

Olympiads in Informatics

17

IOI
INTERNATIONAL OLYMPIAD IN INFORMATICS

ISSN 1822-7732

**INTERNATIONAL OLYMPIAD IN INFORMATICS
VILNIUS UNIVERSITY**

OLYMPIADS IN INFORMATICS

Volume 17 2023

Selected papers of
the International Conference joint with
the XXXV International Olympiad in Informatics
Szeged, Hungary, 28 August–4 September, 2023



OLYMPIADS IN INFORMATICS

Editor-in-Chief

Valentina Dagienė

Vilnius University, Lithuania, valentina.dagiene@mif.vu.lt

Executive Editor

Mile Jovanov

Sts. Cyril and Methodius University, North Macedonia, mile.jovanov@finki.ukim.mk

Ágnes Erdősne Németh

Eötvös Loránd University, Hungary, erdosne@inf.elte.hu

Technical Editor

Tatjana Golubovskaja

Vilnius University, Lithuania, tatjana.golubovskaja@mif.vu.lt

International Editorial Board

Benjamin Burton, University of Queensland, Australia, bab@maths.uq.edu.au

Michal Forišek, Comenius University, Bratislava, Slovakia, misof@ksp.sk

Gerald Futschek, Vienna University of Technology, Austria, futschek@ifs.tuwien.ac.at

Marcin Kubica, Warsaw University, Poland, kubica@mimuw.edu.pl

Luigi Laura, Uninettuno University, Rome, Italy, luigi.laura@uninettunouniversity.net

Ville Leppänen, University of Turku, Finland, villelep@cs.utu.fi

Krassimir Manev, New Bulgarian University, Bulgaria, kmanev@nbu.bg

Seiichi Tani, Nihon University, Japan, tani.seiichi@nihon-u.ac.jp

Peter Waker, International Qualification Alliance, South Africa,

waker@interware.co.za

Willem van der Vegt, Windesheim University for Applied Sciences, The Netherlands,

w.van.der.vegt@windesheim.nl

The journal Olympiads in Informatics is an international open access journal devoted to publishing original research of the highest quality in all aspects of learning and teaching informatics through olympiads and other competitions.

<https://ioinformatics.org/page/ioi-journal>

ISSN 1822-7732 (Print)

2335-8955 (Online)

© International Olympiad in Informatics, 2023

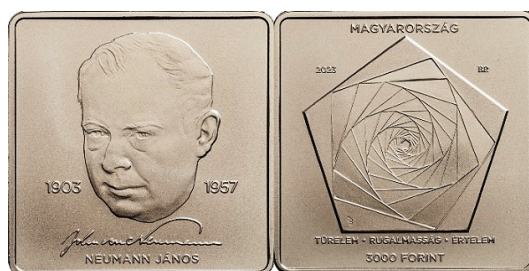
Vilnius University, 2023

All rights reserved

Foreword

The publication of this issue marks a milestone in the International Olympiad in Informatics (IOI): for the first time in three years, the IOI is returning on-site. Alongside it, we will have the first face-to-face IOI conference since 2019.

The IOI will be held in Hungary from August 28th to September 4th, 2023, in commemoration of the renowned mathematician and computer scientist, John von Neumann. This year marks the 120th anniversary of Neumann's birth, and Hungary has organized a series of remarkable events to honour his legacy. The IOI serves as the pinnacle event for the IOI community, among various commemorations: exhibitions, conferences, competitions, scholarships, the Neumann Memorial Coin, and the publication of various books.



Neumann János' collector coin

<https://www.mnb.hu/en/banknotes-and-coins/collector-and-commemorative-coins/2023/collector-coin-to-honour-john-von-neumann>

Hungary, having hosted the IOI once before in 1996, is privileged to have been granted the opportunity once again. The success of the previous IOI, held in Veszprém, under the leadership of the Neumann Society and Eötvös Loránd University (ELTE), has left an indelible mark on our community. This time, a team of professionals from ELTE has prepared the tasks for the competition. Our community at ELTE has an extensive experience in competitive programming, including the development of study materials for talented students, organizing national competitions, selection contests, and preparation camps, as well as being leaders in international Olympiads, we are confident in the quality of the tasks presented.

Participants will be guided by the international student community of the University of Szeged (SZTE, Your Future – Our Mission), where they will have the opportunity to experience the city's hospitality. Moreover, Szeged is home to one of the world's largest collections of historical computers, showcasing the evolution of technology in both Eastern and Western countries. The Neumann Society's experts have diligently expanded and made part of this impressive collection accessible online through the Informatics History Forum (<https://ajovomultja.hu>).

The IOI journal – closely tied to the annual scientific conference held during the IOI – presents the newest research and best practices of computing professionals involved in teaching informatics to talented secondary and high school students. In this 17th volume, we have curated an array of diverse and captivating articles. Firstly, we are delighted to present a unique paper that reminisces about the life and legacy of John von Neumann, enriched with captivating stories and fascinating pictures from his era.

The second part of the volume focuses on research. With distinguished lecturers from ELTE contributing four papers that delve into various topics. Bence Gaál presents unplugged activities with Micro:bits and explores the challenges of robotics activities during a pandemic. László Menyhárt *et al.* analyse an intriguing algorithm related to competitive programming, specifically examining the emergence and mathematical background of variants of the prefix sum. Márton Visnovitz *et al.* discuss current trends in teaching programming in Hungary.

Georgio Audrito *et al.* share their experiences and lessons learned from the introduction of a new talent selection competition. Additionally, Vania Natalia *et al.* conduct an analysis of Indonesian students' computational thinking (CT) knowledge based on their participation in the Bebras challenge, offering insightful ideas for improvement. Pavel Pankov *et al.* present an engaging paper that delves into knowledge beyond the IOI syllabus, highlighting the significance of mathematics, other STEM subjects, and general knowledge required in competitions. They propose time-dependent tasks along with a corresponding time checker. Lastly, Tom Verhoff sheds new light on recursion, exploring its intricacies and providing fresh perspectives on this divisive yet intriguing problem-solving technique.

The third part of this volume features four reports based on national experiences and important news within our community. Michael Dolinsky presents a report on traditional programming Olympiads in the Gomel region for grades 5–8, encompassing motivational aspects. Felix Jingga *et al.* share the impact of changes in the preparation methods of the Indonesian team on their results in the IOI. We also have a report on the national Olympiads in Sri Lanka, followed by Mărtiņš Opmanis *et al.*'s insights into the challenges of a new team competition that allows the use of Internet resources.

Lastly, in the fourth part of this volume, Orit Hazzan *et al.* introduce an innovative and interdisciplinary approach to teaching data science, offering valuable insights into this emerging field.

We would like to express our deepest gratitude to all those who have contributed to this volume, particularly the authors and reviewers. Their dedication and hard work, not only in writing the papers but also in the extensive review and correction process, have been instrumental in producing this exceptional collection. May it be a memorable and enriching experience for everyone involved. We extend our warmest regards to all participants, speakers on the conference, and members of the IOI community, as we eagerly anticipate the upcoming event in Hungary. May it be a memorable and enriching experience for everyone involved.

Editors

As the Epitome of Talent: John von Neumann and Hungarian-born Scientists Around Him

Gábor KÉPES¹, Ágnes ERDŐSNÉ NÉMETH^{1,2}

¹*John von Neumann Computer Society, Budapest, Hungary*

²*Eötvös Loránd University, Budapest, Hungary*

e-mail: kepes.gabor@njszt.hu, erdosne@inf.elte.hu

This paper is dedicated to the lasting memory and impact of an outstanding scholar on the field of computing.

Abstract. John von Neumann (1903–1957) was a Hungarian-American mathematician, physicist, computer scientist, and polymath who made significant contributions to various fields of science and technology. He is best-known for his pioneering work in computer science, game theory, and the development of the Neumann architecture, which forms the basis of most modern computers.

Neumann’s work laid the foundation for the development of the digital computer, and he is often regarded as one of the founding figures of the field of computer science. He made important contributions to the development of early computing machines, including the concept of stored-program computers, which enabled computers to store and manipulate instructions as data, and the idea of self-replicating machines, which has been influential in the field of artificial intelligence and robotics.

In addition to his work in computer science, Neumann made significant contributions to other fields, such as mathematics, physics, economics, and nuclear physics. He also played a key role in the development of the atomic bomb during World War II as part of the Manhattan Project.

Neumann’s work has had a profound impact on modern science, technology, and society, and his legacy continues to inspire and influence researchers and practitioners in various fields. His visionary ideas and interdisciplinary approach to problem-solving make him a remarkable figure in the history of science and technology.

Keywords: John von Neumann, anniversary, talent, CS, digital computer.

1. Introduction

Neumann János – as the international scientific community knows him, John von Neumann – is undoubtedly the most well-known Hungarian mathematician of the 20th century, whom the Financial Times named “Man of the Century” in 1999. In Hungary, he is considered the “father of computers”. Even with less bias, it can be stated that he belonged to the group of computer pioneers – such as the German Konrad Zuse, the British



Fig. 1. A late photo of John von Neumann.

Source: <https://biografieonline.it/foto-john-von-neumann>

Alan Turing, the Americans J. Presper Eckert, John Mauchly, Herman Goldstine, John Vincent Atanasoff – who had a fundamental impact on the history of information technology in the world. They laid the true foundations of modern informatics, building on 19th century precedents such as the algebra of British George Boole and the work of British Charles Babbage and Ada Lovelace in the development of programmable machines.

Describing the principles of operation of the modern computer and making it publicly available was just a small part of Neumann's extensive work, as his thoughts enriched numerous scientific fields during his fifty-four years of life.

In Hungary, Neumann is held in extraordinary respect. The John von Neumann Computer Society (NJSZT) – which adopted its name 55 years ago – has since become a well-established institution among Hungarian computer scientists, recognizing his significant and outstanding role in the evolution of computers. It is particularly interesting that Neumann was well-respected in Hungary even during the cold war, even though he had emigrated to the USA.

NJSZT has been involved in international talent nurturing programs from the very beginning, as well as in the community of competitive programming Olympiads, establishing a solid connection between von Neumann, Hungary, and the international network of teachers for decades.

This year marks the 120th anniversary of Neumann's birth, which is the occasion for Hungary to host the International Olympiad in Informatics in Szeged, where the IT Museum of the NJSZT is also located. The museum has been greatly influenced by the intellectual legacy of the University of Szeged, as well as the spirit of another computer pioneer, László Kalmár. The exhibition prominently presents the life's work of Neumann. In honour of the 120th anniversary of Neumann, the NJSZT has planned numerous celebratory programs for 2023, including a traveling exhibition based on the panels of our current study.

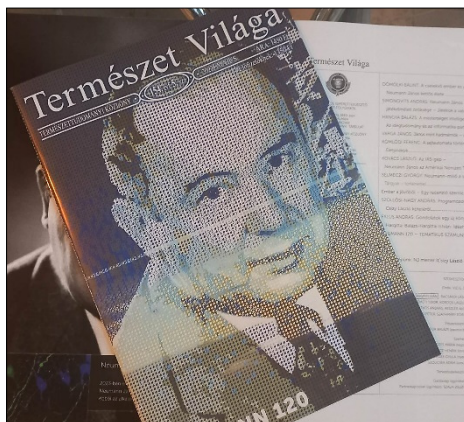


Fig. 2. The exclusive issue of “Természet Világa” magazine dedicated to Neumann.
Photo: njszt . hu

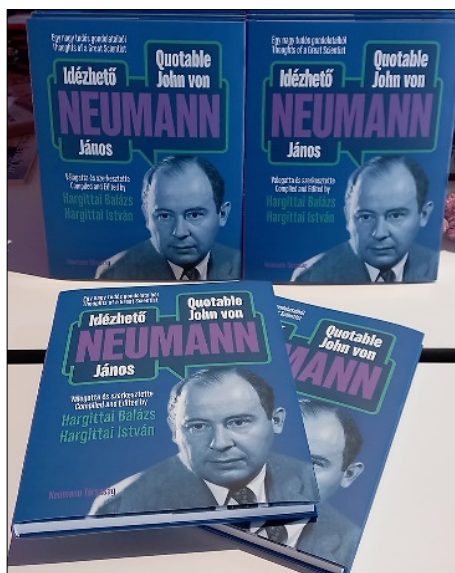


Fig. 3. A new bilingual (Hungarian-English) volume titled “Quotable John von Neumann”
has been released by NJSZT. Photo: njszt . hu

2. Neumann, the Talent. The importance of Family Environment and Education

The first third of the 20th century saw the birth of several great scientists in Budapest, including Theodore von Kármán, a pioneer of rocket technology, Leo Szilard, Eugene Wigner, and Edward Teller, giants of nuclear physics, Neumann, the father of the computer, and John G. Kemeny, co-author of the BASIC language.



Fig. 4. Interior from the exhibition “Neumann Milieu” by NJSZT.
The paintings depict John von Neumann’s aunt and maternal grandmother.
The furniture belonged to the Neumann family. Photo: njszt.hu

Neumann was born on December 28, 1903, and his childhood home was in the city centre of Budapest (at 62 Bajcsy-Zsilinszky Street). He came from a wealthy and highly educated Jewish family. His father, Miksa Neumann, was a respected banker who was granted a noble – baronial – title by Emperor Franz Joseph I and took on the prefix “of Margitta”. According to family memories, the name Neumann referred to János’ mother, Margit Kann, and the daisy, which symbolized her. János had two brothers, Mihály and Miklós. They grew up in great love, and the stained-glass window in the Neumann House (made by Miksa Róth, the most famous Hungarian glass artist) reminds us of all three of them: János as the rooster, Mihály as the rabbit, and Miklós as the cat.

It was typical of the family environment that famous scholars were guests at their dining table, and family members sometimes joked with each other in Ancient Greek. According to the summary of Endre Czeizel, the geneticist professor, Neumann had already mastered differential and integral calculus at the age of eight. Lipót Fejér and Rudolf Ortvy, mathematician/physicist professors, also visited them frequently. They also had dinner guests such as Frigyes Karinthy, the writer, Dezső Kosztolányi, the poet and Max Reinhardt, the theatre director. (Czeizel, 2011)

“Genius training” of early childhood continued in high school (Budapest Ágostai Lutheran High School), where Neumann became student of László Rátz. His other students included Eugene Wigner, the future Nobel Prize-winning physicist, and John Harsanyi, later Nobel Prize-winning economist studied in this community.

Jancsi – as Neumann was called during his childhood – received special attention from his teacher, László Rátz. They edited a mathematical journal together in high



Fig. 5. Selmeczi giving a guided tour at the “Neumann Milieu” exhibition. Above him are the paintings by Cézár Kunwald depicting John von Neumann’s maternal grandfather (pointing at him) and teacher László Rátz. Photo: njszt.hu

school while others progressed with normal curriculum. His teacher organized meetings with professors from the Technical University for the exceptionally talented boy.

Later László Rátz became Hungary’s most famous talent developer. Since 2000, “Rátz László Teacher Lifetime Achievement Award” recognizes the most significant high-school teachers in mathematics, physics, biology, and chemistry in Hungary.

In 2023, NJSZT presented an exhibition titled “Neumann Milieu” at his high school with the original furniture of Neumann’s family. The interior designer György Selmeczi – who owns the objects – discovered that the paintings in his possession depict Neumann’s maternal grandparents. The portraits of Neumann’s grandfather, Jakab Kann, and his teacher, László Rátz, were placed side by side at the exhibition. Both oil paintings were created by Hungarian painter, Cézár Kunwald. According to Selmeczi’s assumption, Neumann’s father supported the school by immortalizing the teacher with his “house painter”. (Selmeczi, 2023)

John von Neumann was reserved but had a good sense of humour and expressed himself wittily. This is evident from the letters he wrote to his classmates. (Szabó)

He studied mathematics, experimental physics, and chemistry in Budapest, and philosophy, mathematics, physics, and chemistry in Berlin. At the request of his father, the young man with a fundamental interest in mathematics also obtained a “useful” chemical engineering degree in Zurich, Switzerland, but he obtained his doctorate in mathematics in Budapest.

John von Neumann followed the development of Hungarian science, and even during his lifetime, he was esteemed by Hungarian scholars. However, during the era of the infamous “Jewish laws”, World War II, and the Cold War, he became a prominent figure in the scientific community of his adopted country, the United States of America. He visited Budapest several times until World War II.

His first wife was Marietta Kövesi, they had a daughter, Marina von Neumann-Whitman (Marina later became a renowned economist and advisor to President Nixon). In 1938 Neumann divorced from Marietta and married Klára Dán, who was one

of the world's first programmers and a perfect intellectual partner for him. They emigrated together to the United States.

3. Neumann, the Scientist Role Model and Gamer

After completing his university studies in Budapest, Neumann taught as a private lecturer at the University of Berlin (1926–28) and at the University of Hamburg (1929–1930). Between 1930 and 1933, he taught at Princeton University. In 1933 he was invited to the renowned Institute for Advanced Study (IAS) in Princeton, where some of the world's most distinguished scientists – including Albert Einstein and Kurt Gödel – worked.

The pure mathematics fascinated and occupied him throughout his life. Between 1926 and 1937, he focused mainly on mathematical and quantum mechanical questions.

In 1928, the minimax principle was formulated in connection with two-player, zero-sum games, stating that one should choose the option that minimizes the maximum loss. This gave birth to game theory, which evolved from a branch of mathematics and the observation of card games, board games, sports – including poker, favoured by Neumann. Game theory assumes rational decision-making (utility maximization) and the interactions it observes are not only present in games, but also in arms races and wars as well as many other areas (economics, politics, psychology, sociology, ...)

After contributing to the axiom system of set theory, he embarked on axiomatizing quantum mechanics. As a result of his work, his fundamental book, “The Mathemati-



Fig. 6. John von Neumann in his youth, around the 1920s.
Source: Archive of the Budapest-Fasori Lutheran High School.

cal Foundations of Quantum Mechanics”, was published in 1932. His further research, together with G. Birkhoff, proved that quantum mechanics requires a different logic than classical mechanics.

In 1944, Neumann published the book “Theory of Games and Economic Behaviours” with Oskar Morgenstern, making it clear how the examination of games can be useful in economics. The book was a milestone for psychology, economics, political science, and history, as their foundations – in many ways – lie in the concept of a game. Most researchers agree that game theory connects social sciences and is a great help in the analysis of human interactions.

John von Neumann did indeed have a deep love for games – as mentioned by his daughter in her autobiography, stating that her father always kept games at his fingertips. (Neumann-Whitman, 2012) If he had lived during the era of computer games, he would likely have become an avid gamer. From this perspective, he is also suitable as an attractive role model for children: not a recluse avoiding enjoyable activities, but a true humorous and playful mind.

4. Neumann and Historical Responsibility

In 1937, Neumann obtained US citizenship. By this time, World War II seemed inevitable, and he eventually became involved in military preparations against Nazism. His interest shifted increasingly towards the practical applications of mathematics. As an advisor, he participated in several US military projects. From 1943 he regularly visited Los Alamos, where he was involved in the theoretical and practical work related to the development of the first atomic bombs.

The secret program was the Manhattan Project, which had its starting point in a letter from Albert Einstein to President Roosevelt in 1939, initiated by Leo Szilard and Eugene Wigner. In the letter, Einstein strongly emphasized that new research indicated that a new, extraordinarily powerful bomb could be created from uranium. Einstein pointed out in the letter that Nazi Germany was also intensively engaged in similar research, and



Fig. 7. Eugene Wigner. Source: wignerkozepiskola.hu

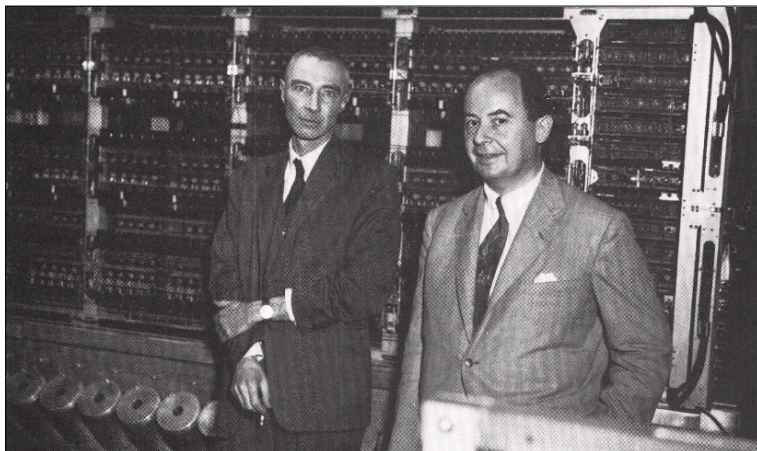


Fig. 8. J. Robert Oppenheimer and John von Neumann – standing in front of computer Alan Richards, photographer, 1952. From the Photograph collection. Shelby White and Leon Levy Archives Center, Institute for Advanced Study in Princeton, NJ.

as a result, the Third Reich had already halted the export of mined uranium in occupied Czechoslovakia.

Alongside Robert Oppenheimer, Enrico Fermi, Leo Szilard, Edward Teller, and Eugene Wigner became key figures in the project. At that time, Hungarian-born scientists of this generation were nicknamed „Martians” – most likely because they spoke to each other in Hungarian, which sounded exotic to American ears, and because their knowl-



Fig. 9. Edward Teller, in 1958, as Director of Lawrence Livermore National Laboratory.

Source: [https://commons.wikimedia.org/wiki/File:EdwardTeller1958_\(dust_%26_scratches\).jpg](https://commons.wikimedia.org/wiki/File:EdwardTeller1958_(dust_%26_scratches).jpg)

edge was almost „otherworldly.” Moreover, Edward Teller’s monogram was E.T., which stood for aliens. (Marx, 1997)

During the study of shock waves generated by the detonation of atomic and hydrogen bombs, Neumann discovered complex mathematical relationships that could not be solved using classical methods. This led him to become interested in the possibilities of high-speed electronic computing. In the United States, the scientist – known as John von Neumann – participated in the atomic program, despite knowing the dangers and moral objections. Meanwhile, in the 1940s and 1950s, he became a leading expert and a prominent public figure considered as an opinion leader by U.S. presidents.

We consider it is very important what Marina, her daughter said at the opening of NJSZT’s Informatics History Exhibition in Szeged: “...my father led a dual life: as a leading figure in the ivory tower of pure science, and as a man of action whose advisory and decision-making activities were constantly in demand in the long struggle to ensure that the United States would prevail in both the hot and cold wars that dominated the half-century from 1939 to 1989.” (Neumann-Whitman, 2013)

He participated in the research and military application of nuclear energy and played a role in the direction of peaceful energy production: In 1954, he was appointed as a member of the five-member Atomic Energy Commission (AEC) of the USA.

5. The Father of Modern Computers

Although many people mistakenly refer to Neumann as the inventor of the computer, he was not an inventor. What we now call a computer was the result of a series of innovations. Major milestones: (Dömölki, 2016)

- 1941: Zuse, Z3: stored-program, relay-based computer.
- 1942: Atanasoff, ABC: electronic purpose-built machine.
- 1944–46: Mauchly-Eckert, ENIAC: electronic, general-purpose machine.
- 1944–45: EDVAC a stored-program, electronic, general-purpose machine.

In 1944, Neumann met Herman Goldstine at the Aberdeen train station, who was one of the leaders of the ENIAC construction and Goldstine told him about developing an electronic structure with enormous computing power. Neumann’s eyes lit up: he was revered in his circle as an almost fearsomely knowledgeable “human computer”, but for certain mathematical, physical, and military problems – even the combined capacity of a whole village of mathematicians, mechanical calculators, and years of work – were not enough.

In the fall of 1944, Neumann joined the designers of the EDVAC – a new computer to be built based on the experiences with the ENIAC – at the Moore School (University of Pennsylvania, Philadelphia, USA). Based on the results of their collaborative work, he formulated the operational principles of the EDVAC in his “First Draft on a Report of EDVAC”. In this innovative approach to computer design, he provided a plan for the logical structure of the machine – instead of describing the hardware components – al-



Fig. 10. Herman Goldstine at the presentation of the Hungarian edition of the book “The Computer from Pascal to von Neumann,” standing next to him is Győző Kovács, the Secretary-General of NJSZT, in 1987. Photo: NJSZT.

lowing for its realization in various hardware environments. The principles formulated by Neumann in this document:

- Fully electronic computer.
- Use of binary number system.
- Application of arithmetic unit.
- Application of central control unit.
- Internal program and data storage (the principle of stored program).

The first electronic computers – which were as big as a room – generated a lot of heat, had high energy requirements, and required frequent repairs, were initially custom-made machines. Neumann foresaw their military and scientific applications, but their widespread adoption – due to their size and cost – was not evident for a long time.

Neumann himself took on the direction of designing a new stored-program computer: this machine, the famous IAS, was completed by 1951 in Princeton. It fully met all the requirements of the Neumann’s architecture. The scientist – who enjoyed hosting large parties and social events – threw a party to celebrate the unveiling of the machine: the highlight of the evening was the presentation of an ice sculpture model of the computer, alongside fine martinis. While the ice sculpture melted, the “grandchildren” of the IAS machine – referring to the spread of similar computer designs around the world – became widely adopted. (Neumann-Whitman, 2012)

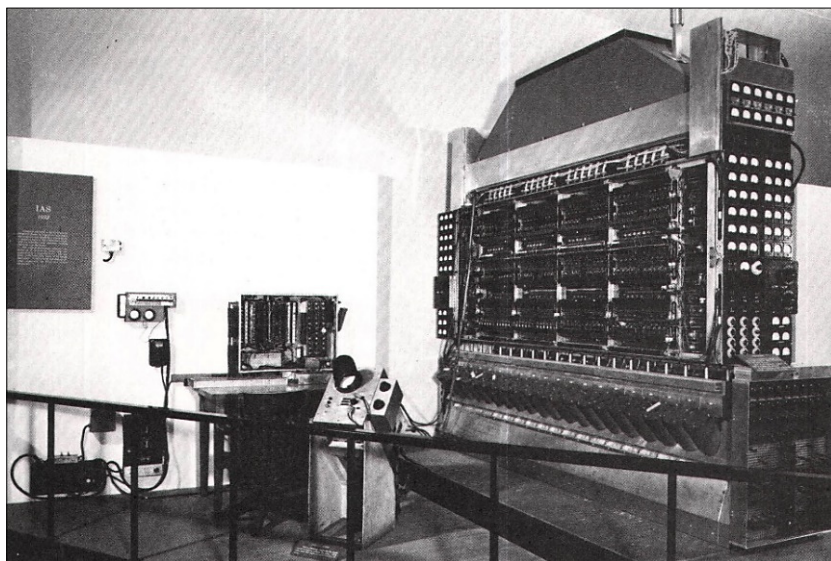


Fig. 11. Princeton IAS Computer, the complete system.

Source: <https://historyofinformation.com/image.php?id=5524>

The most advanced Neumann machine, the IAS, weighed nearly half a ton and utilized 1700 vacuum tubes. It had a memory capacity of approximately 5 kilobytes! with word lengths of 40 bits. It could perform 16,000 additions and 400 multiplications per second. This computer – which incorporated numerous innovations such as the efficient use of Williams tubes as memory and small oscilloscope screens as displays – remained in operation until 1958. It was relatively reliable compared to other computers of the time and was used for calculations in fields such as nuclear physics and numerical meteorology. (Aspray, 2004)

In Hungary – the first computer based on Neumann's principles – the M-3 was completed in 1959, based on a Soviet design, refined and further developed by the Cybernetics Research Group of the Hungarian Academy of Sciences (MTA KKCS). The day of the presentation of the M-3 is proposed by NJSZT as a new celebration: the Hungarian IT Day is 21st of January. Rezső Tarján, the deputy director of MTA KKCS, and Győző Kovács, the head of the first computer center based on the M-3, became dedicated supporters and promoters of Neumann's intellectual legacy.

Neumann considered the computer as a new scientific achievement of human progress. Therefore, he was not opposed to the emergence of new computers based on the refined Neumann principles and – if his health allowed – he followed the mathematical and programming challenges arising from these computers. "The genius of John von Neumann not only facilitated the birth of high-speed computers, but also the solution of mathematical problems, arising from their utilization," characterized the significance of the scientist by Hungarian mathematician, Gyula Obádovics. (Obádovics, 2003)

In the over half a century of digital computers the Neumann principles have been defining, whether it be the behemoth machines of the early days or the smart pocket



Fig. 12. Electron tubes from the first Hungarian von Neumann-inspired computer, M3.
Photo: ajovomultja.hu

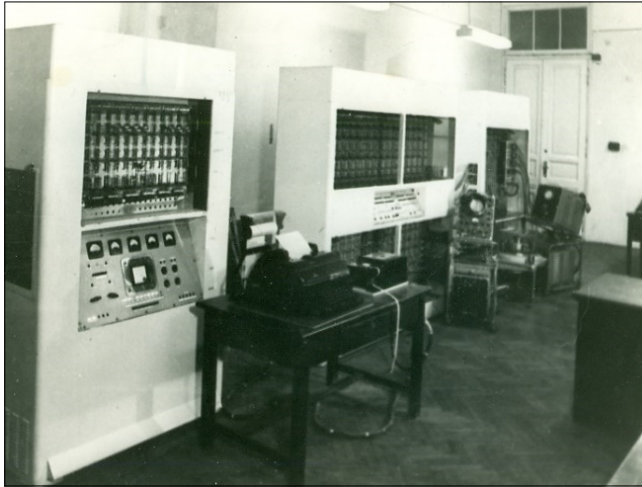


Fig. 13. The first Hungarian von Neumann-inspired computer, the M-3, in 1959.
Photo: ajovomultja.hu

devices of today. Of course, classical computer design principles are now supplemented with new knowledge, as we are living in the age of microelectronics and nanotechnology, and we may even be witnessing the dawn of quantum computing.

6. The Antechamber of Artificial Intelligence

“What kind of logical structure is sufficient for an automatic machine capable of self-reproduction?” Neumann posed the question. As a true mathematician, he also contemplated computers on a theoretical level. In early 1940s, he introduced the concept of

“cellular automata,” a mathematical model in which cells can take on different states. He developed a “Universal Constructor” with cells capable of assuming 29 different states. The main theme of his 1948 lecture titled “The General and Logical Theory of Automata” was the formulation of new logical principles for automata, including self-replicating automata. His last unfinished (posthumously published) masterpiece, “The Computer and the Brain,” is about computers as automata. It aims to approach the understanding of the nervous system. He compares neurons and neural memory with artificial components, seeking analog and digital parallels.

Neumann stated: “In the future, science will be more concerned with the problems of regulation and control, programming, data processing, communication, organization, and system management.” He recognized that the security and efficiency of a system are not determined by the elements it is composed of, but by how it is organized as a system and the quality and quantity of information that flows between the elements.

Since Neumann’s ideas, the field of cellular automata has undergone significant theoretical development, and automation has evolved into a vast area encompassing robotics and artificial intelligence. A successful Hungarian entrepreneur, Gábor Bojár compares the revolution of computers to the emergence of speech and writing in human history: “but even those who express more cautious views acknowledge that we are living in an era of another industrial revolution, the foundations of which were laid by Neumann and his generation.” (Bojár, 2022)

Indeed, Neumann played an incredibly significant and pioneering role in the development of cellular automata theory, which has had a profound impact on various fields such as life games (popularized by John Conway), evolutionary genetics, and the concept of self-replicating automata. His principles for creating reliable machines from unreliable components also shape the entire IT industry, and his propositions regarding



Fig. 14: The wax figure of John von Neumann exhibited in 2023 at the Madame Tussauds, Budapest. Photo: njszt.hu

the functioning of the human mind can serve as inspiration for both IT professionals and brain researchers alike. These fields are not so far apart, and Neumann's work has fundamental importance in many areas of IT.

7. Can we Survive Technology? Neumann, the Visionary

John von Neumann passed away in 1957, suffering from bone cancer, likely because of radiation exposure. He was a highly respected personality whose merits were recognized in various ways during his lifetime. He was a member of the National Academy of Sciences, President of the American Mathematical Society, and received the Medal of Freedom from the President of the United States (1956) – which Eisenhower presented to him at his hospital bed – as well as the Albert Einstein Medal and the Enrico Fermi Award from the U.S. Atomic Energy Commission (1956).

His legacy left to the world was of great interest. He was not even sure if what he had created would still be interesting “a hundred years from now.” Immortality was granted to him through his theoretical mathematical summations and his practical contributions with real-world impact.

We owe him a lot in terms of weather observation as well. He was very interested in the role of computers in weather forecasting and even in climate modification. He put the ENIAC computer into service of numerical meteorology and the accurate prediction of weather. By numerically integrating the barytropic vorticity equations, he achieved the first successful 24-hour forecast for North America based on actual data from four selected days in the first two months of 1949. He continued to work on this topic with his later machines, and even organized conferences on computer modelling of climate processes.

In 1955, Neumann wrote an almost prophetic article titled “Can we survive technology?” for Fortune magazine. Although Neumann was not yet a “climate alarmist” (since the term did not exist at the time), he accurately foresaw the increasing globalization,



Fig. 15. József Füzér's caricature of John von Neumann.
Photo: njszt.hu

interdependence of humanity, and the opportunities presented by nuclear energy and automation, and treated the possibility of weather control as a fact. “Technologies are always constructive and beneficial, directly or indirectly. Yet their consequences tend to increase instability...” (Neumann, 1955)

He believed that the impacts of the achievements created by his generation were explosive in nature, and their magnitude was “the size of the whole world.” “For progress, there is no cure” he said, meaning that progress cannot and should not be stopped. The only thing we can do is to intelligently execute daily decisions.

“The one solid fact is that the difficulties are due to an evolution that, while useful and constructive, is also dangerous. Can we produce the required adjustments with the necessary speed? The most hopeful answer is that the human species has been subjected to similar tests before and seems to have a congenital ability to come through, after varying amounts of trouble. To ask in advance for a complete recipe would be unreasonable. We can specify only the human qualities required: patience, flexibility, intelligence.” (Hargittai *et al.*, 2023)

It’s dizzying to think about all the interesting questions we could have read Neumann’s thoughts on, had he lived for almost 100 years, like his brother Miklós. What would he have said about the information society? Personal computers, video games, mobile phones? The end of the Cold War? The ecological crisis, climate change?

Neumann could have been a true role model for the generation facing the great challenges of the 21st century, the current students of Olympiads, whose task is to solve challenges determine the future of humanity. The stakes are high in nurturing this talent, as we search for the future Neumanns.

References

- Aspray, W. (2004). Neumann János és a modern számítástechnika kezdetei, Vince Kiadó.
- Bojár, G. (2022). 4th industrial or 3rd IT revolutions? Speech at Neumann Conference.
<https://www.youtube.com/watch?v=AbsCmPgmVVQ>
- Czeizel, E. (2011). Matematikusok – gének – rejtélyek, Galenus Kiadó.
- Dömölki, B. (2016). John von Neumann in Computer Science, The 2016 IEEE International Conference on Systems, Man, and Cybernetics.
https://njszt.hu/sites/default/files/page/2018/domolki_-_john_von_neumann.pdf
- Hargittai, B., Hargittai, I. (2023) Quotable John von Neumann, NJSZT
- Kovács, Gy. (1997). Neumann János, Műszaki Kiadó
- Kovacs, Gy. (n.d.). Hungarian Scientists in Information Technology in Reflections on the History of Computing (Ed. Arthur Tatnall), IFIP AICT I.
<https://dl.ifip.org/db/series/ifip/ifip387/Kovacs12.pdf>
- Marx, Gy. (1997). A Marslakók legendája, Fizikai Szemle, XLVII./3.
<https://mek.oszk.hu/03200/03286/html/tudos1/marsl.html>
- Neumann-Whitman, M. (2012). *The Martian’s Daughter: A Memoir*. The University of Michigan Press
- Neumann-Whitman, M. (2013). The Past of the Future Conference – Keynote speech.
<https://www.youtube.com/watch?v=MbAtnVgZ1LU>
- Neumann, J. (1955). Can we survive technology? *Fortune Magazine*.
<https://fortune.com/2013/01/13/can-we-survive-technology/>
- Neumann120 (2023). *To celebrate the 120th anniversary of von Neumann’s birth, the John von Neumann Society is preparing a series of commemorative events. Our aim is to make the von Neumann heritage as accessible as possible. Join in with us!* NJSZT. <https://n120.njszt.hu/>

NJSZT History of Informatics: <https://itf.njszt.hu/szemely/neumann-janos-john-von-neumann>
 NJSZT's Repository preserves the memoirs of Neumann about his brother, Miklós.

<https://itf.njszt.hu/objektum/john-von-neumann-as-seen-by-his-brother>

Obádovics, Gy. (2003). Az első számítógép alkalmazásával megjelenő numerikus problémák, In: Ki volt igazából Neumann János?, *Nemzeti Tankönyvkiadó*, p. 5–40.

Selmeczi, Gy. (2023). Neumann-milió a szülői házban, *Természet Világa*, V. p. 228–233

Szabó, Zs. cared Neumann's letters in the archiv of Fasori High School



G. Képes obtained an MA degree in Hungarian language and literature at Faculty of Humanities, Eötvös Loránd University in 2004, and also pursued studies in digital humanities at the same institution. Between 2004 and 2014, he served as the curator of the computing collection at the Hungarian Museum of Science, Technology and Transport. From 2014 to 2016, he was the head of department at the Hungarian National Digital Archive. Since 2016, he has been a senior researcher at the Neumann Society (NJSZT), and he has been the director of marketing since 2021. In 2014, he was awarded the Janos Kemény Prize for his publications on the history of informatics.



Á. Erdősné Németh is an assistant professor at Faculty of Informatics, Eötvös Loránd University in Hungary. Previously, she taught mathematics and informatics at Batthyány Lajos High School in Nagykanizsa, where many of her students excelled in national programming competitions and some of them in CEOI and IOI. Her research focus is on promoting computational thinking among all students and preparing talented pupils for competitive programming contests, in primary and secondary schools. Since 2018, she has been serving as the vice-president of Neumann Society (NJSZT).

Giochi di Fibonacci: Competitive Programming for Young Students

Giorgio AUDRITO¹, Madalina CIOBANU², Luigi LAURA³

¹*Department of Computer Science, University of Torino, Italy*

²*University of Molise, Italy*

³*Uninettuno University, Rome, Italy*

e-mail: giorgio.audrito@unito.it, madalina.ciobanu@unimol.it,

luigi.laura@uninettunouniversity.net

Abstract. In this paper, we share the experience gained by organizing and running a programming contest for upper primary and lower secondary schools students; contestants compete in their own age division. This contest, called *Giochi di Fibonacci* (Fibonacci’s games), is organized in three phases, where the first one is based only on logical and algorithmical quizzes, whilst the other two deal with coding, either in Scratch or in a simplified pseudo-code programming environment developed for this scope.

Keywords: programming contest, Olympiads in Informatics, peer education, programming training.

1. Introduction

The integration of computational thinking skills and computer programming in primary and lower secondary education has become increasingly important in recent years, as also witnessed by the large success of Bebras¹ (Dagienė, 2008); see also the recent work of Dagienė *et al.* (2022) for a picture of the worldwide diffusion of Computational Thinking teaching in primary schools, whilst some considerations about the curriculum and teachers’ perspectives related to the introduction of informatics in primary education are discussed in (Dagienė *et al.*, 2019).

Indeed, as also observed in (Dolinsky, 2022), there is a plethora of approaches: unplugged education (Vegt, 2016; Pluhár, 2021), the use of Scratch (Fagerlund *et al.*, 2020), robot programming (Kanemune *et al.*, 2017), LEGO robotics (Souza *et al.*, 2018), and gamification (Combéfis *et al.*, 2016).

In this context, programming contests can serve as an effective tool to motivate students and expose them to problem-solving challenges. In this paper, we describe the introduction

¹ <https://www.bebas.org/>

of a programming contest in upper primary and lower secondary education, with each age division competing separately, aimed at enhancing students' computational thinking and programming skills. The competition, named "Giochi di Fibonacci" (Fibonacci's games), is comprised of three distinct stages, where the initial stage is solely based on logical and algorithmic assessments, similar to Bebras, while the other two phases involve the use of coding, either via Scratch or a specially designed simplified pseudo-code programming environment catered to this competition.

In the literature, there are reports about similar experiences (in Belarus (Dolinsky, 2022)) and also about training students for this events (Vegt, 2016; Kiryukhin *et al.*, 2022).

This paper is organized as follows: in the next section we provide a general overview of the *Giochi di Fibonacci*, whilst the three sections that follow detail the three phases of the competition, providing info about both the structure and the results obtained. Then, in Section 6, we discuss the lessons we learnt in this first experimental edition, together with some changes we plan to implement in the next year. Finally, Section 7 addresses final remarks and conclusions.

2. Giochi di Fibonacci

Our competition mimics the structure of the Italian Olympiads in Informatics (Audrito *et al.*, 2021), divided into three phases as well. The first phase does not involve coding, and problems proposed are similar to the ones of Bebras, thus aiming to involve students from that competition to participate.

Why Fibonacci. Leonardo Pisano, more commonly known as Fibonacci, is a mathematician from the 13th century. He learned the Hindu-Arabic numeral system during his travels to North Africa with his father, a customs agent, and described them in the *Liber Abaci*, or "Book of Calculation", which revolutionized the way commerce was conducted. It enabled average individuals to buy and sell goods, convert currencies, and maintain accurate records of their possessions more easily than ever before. Its publication led to extensive international commerce and contributed to the scientific and artistic advancements of the Renaissance.

Fibonacci is most famously known for the Fibonacci sequence, which is a recurring pattern of numbers that can be generated using a simple recursive algorithm. This algorithm is based on the idea that each number in the sequence is the sum of the previous two numbers:

$$fib(n) = fib(n - 1) + fib(n - 2)$$

where the first two numbers in the sequence are 0 and 1, and thus the sequence begins with 0, 1, 1, 2, 3, 5, 8, 13, 21, etc. In his book (Devlin, 2011), Devlin points out the remarkable similarities between the computing revolution that took place in Tuscany during the 13th century, under the guidance of Fibonacci, and the one that began in California's

Silicon Valley more recently, with the personal computing revolution of the 1980s started by Steve Jobs, the founder of Apple computers, with the introduction of the mouse and a graphical interface. Devlin offers a unique perspective, showing how history repeated itself.

The structure of the competition The competition has been organized in three distinct phases. Due to the young age of the participants all the phases took place in their own schools, under the supervision of their own teachers.

- *First phase*: logical and algorithmic quizzes, similar to Bebras but with more weight on “program reading” quizzes.
- *Second phase*: pseudo-code or Scratch programming.
- *Third phase*: pseudo-code or Scratch programming, with more difficult problems.

In the following sections we discuss each of the phases, describing in detail the types of exercises proposed and the overall feedback received after the conclusion of this first experimental edition.

3. First Phase

During the first phase, students could access the administered exercises through appropriate accessible online tools both via computer and via tablet or smartphone². The exercises provided were intended to measure logic and basic mathematics thinking, ability to identify problem-solving algorithms and ability to understand descriptions of simple procedures. In this phase, no knowledge of programming languages was required to carry out and understand the exercises.

Questions required either multiple-choice or numeric answers. All multiple choice questions had 5 options, of which only one was correct. The score assigned for these questions was:

- 5 points for a correct answer.
- 1 point for a blank answer.
- 0 points for an incorrect answer.

Each numerical open-ended question required an integer (possibly negative) number as an answer. The score assigned for these questions was:

- 5 points for a correct answer.
- 0 points for an incorrect or blank answer.

The phase was managed and carried out independently by every single educational institution, at the times most suited to them, giving 50 minutes to students to complete the test. Further details follow, divided between primary and lower secondary schools. Overall, we had 9 questions for primary school and 10 questions for lower secondary school, and the response distributions are shown, respectively, in Fig. 1 and Fig. 2.

² An interactive training version of the exercises is available online at: <https://scolastiche.olinfo.it>

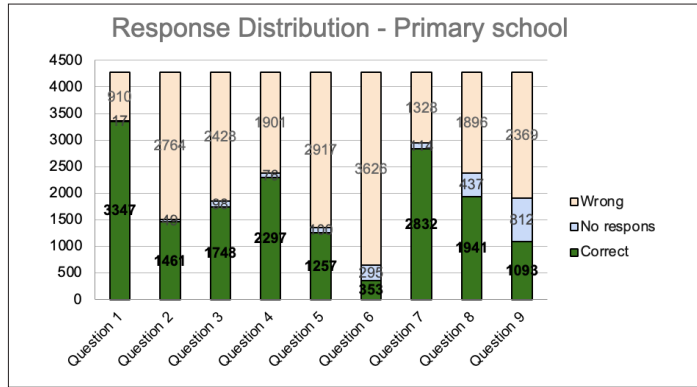


Fig. 1. Response distribution in primary school.

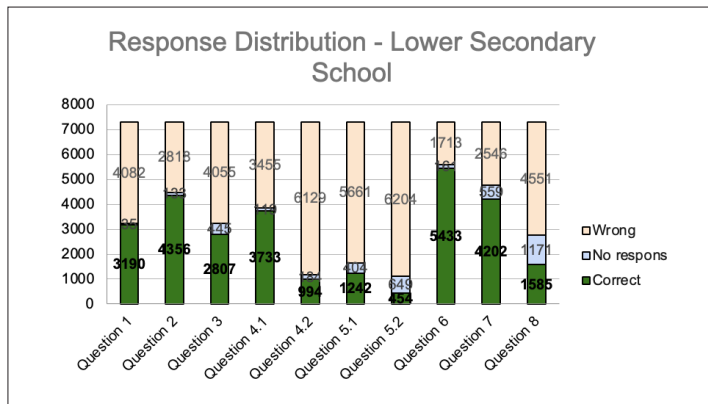


Fig. 2. Response distribution in lower secondary school.

3.1. Primary School

The test for primary school contained 9 questions, divided into three parts as follows:

- Logical thinking (4 multiple-choice questions).
- Algorithmic thinking (2 open-ended numeric questions).
- Program reading (3 multiple-choice questions).

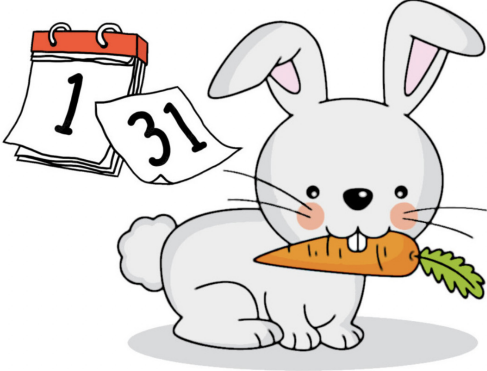
In each of the three parts, the questions were roughly ordered by increasing difficulty. A sample of the questions follows.

Question 1. *Every Monday, Tap-Tap picks 10 carrots from his garden. Every day of the week, Tap-Tap eats a carrot. How many carrots does he have left over each week?* (see Fig. 3)

Question 6. *Tip-Tap has gone on a trip to Turing's farm, and he wants to bring back lots of carrots to his farm mates. On Turing's farm, the Carrot Market takes place every*

week, where it is possible to buy many boxes of carrots, each at a cost of 10 carrots. Carrots are good for your eyesight, and Tip-Tap eats so many he can see through the boxes! These are the numbers of carrots contained in each box [. . .] how many carrots can Tip-Tap earn at most, by buying a set of boxes of his choice? (see Fig. 4)

Ogni lunedì, Tap-Tap raccoglie 10 carote dal suo orto. Ogni giorno della settimana, Tap-Tap mangia una carota. Quante carote gli avanzano ogni settimana?



Domanda 1

- ☐ (A) 10
- ☐ (B) 3
- ☐ (C) 7
- ☐ (D) 4
- ☐ (E) Nessuna

Termina e controlla

Nessuna risposta

Fig. 3. User interface for question 1 (multiple-choice, logic thinking).

Tip-Tap è andato in viaggio alla fattoria di Turing, e vuole riportare tante carote ai suoi compagni di fattoria. Nella fattoria di Turing ogni settimana si svolge il Mercato delle Carote, nel quale è possibile comprare tante scatole di carote, ognuna al costo di 10 carote. Le carote fanno bene alla vista, e Tip-Tap ne mangia così tante che riesce a vedere attraverso le scatole! Questi sono i numeri di carote contenuti in ogni scatola:

1	20	4	11	9	18	12	9	10	11	8	2	3	17	15	13
---	----	---	----	---	----	----	---	----	----	---	---	---	----	----	----

Quante carote può guadagnare al massimo Tip-Tap comprando tutte le scatole che vuole?

Domanda 6

La tua risposta:

Fig. 4. User interface for question 6 (open-ended, algorithmic thinking).

This question is the one that received the most wrong answers as we can see from the graph in Fig. 1. That was somewhat expected, as the topic of the question (algorithmic thinking) is mostly novel for this age group. In primary schools in Italy, competitions already exist that develop logical thinking, but not developing algorithmic thinking.

Question 8. Consider this process, represented as a flowchart. The procedure refers to three numerical variables, represented by letters a , b and c . This program runs twice. The first time the variables are assigned values $a = 7$, $b = 4$ and $c = 6$. The second time the values assigned are instead $a = 5$, $b = 7$, $c = 9$. What numbers does the procedure write in the two runs? (see Fig. 5)

Considera questo procedimento, rappresentato come diagramma di flusso. Il procedimento si riferisce a tre **variabili** numeriche, rappresentate dalle lettere a , b e c .

```

graph TD
    Inizio([Inizio]) --> D1{a è più grande di b?}
    D1 -- SI --> D2{a è più grande di c?}
    D1 -- NO --> D3{b è più grande di c?}
    D2 -- SI --> W1[scrivi il valore di a]
    D2 -- NO --> W2[scrivi il valore di c]
    D3 -- SI --> W3[scrivi il valore di b]
    D3 -- NO --> W4[scrivi il valore di c]
    W1 --> Fine([Fine])
    W2 --> Fine
    W3 --> Fine
    W4 --> Fine
  
```

Questo programma viene eseguito due volte:

- La prima volta vengono assegnati alle variabili i valori $a = 7$, $b = 4$ e $c = 6$.
- La seconda volta vengono invece assegnati i valori $a = 5$, $b = 7$, $c = 9$.

Quali numeri scrive il procedimento?

Domanda 8

☐ (A) 7 e 9

☐ (B) 6 e 9

☐ (C) 7 e 7

☐ (D) 5 e 7

☐ (E) 4 e 5

☒ Lascia in bianco

Fig. 5. User interface for question 8 (multiple choice, program reading).

3.2. Lower Secondary School


The test administered to lower secondary school students consisted of 10 questions, divided into three parts as follows:

- Logical thinking (3 multiple-choice questions).
- Algorithmic thinking (4 open-ended numeric questions).
- Program reading (3 multiple-choice questions).

In each of the three parts, the questions were roughly ordered by increasing difficulty. Some of the easier questions were shared with the test for primary school: 2 logical thinking questions, 2 algorithmic thinking questions, and all program reading questions. The two additional algorithmic questions were increased difficulty follow-ups of the two questions shared with primary schools. A sample of the questions follows.

Question 1. *Bunny found three piles of books in the library of the Fibonacci farm! Bunny would like to read any two gardening books, and he knows that the books on this subject are the ones with a yellow cover. To get a book Bunny has to move all the books above it. Bunny is very lazy, so he wants to be able to grab any two of the books he's interested in by moving as few books as possible! What's the minimum number of books Bunny has to move, counting the gardening books he takes? (see Fig. 6)*

Bunny ha trovato nella libreria della fattoria Fibonacci tre pile di libri!



Bunny vorrebbe leggere due libri di giardinaggio qualunque, e sa che i libri su questo argomento sono quelli con una copertina gialla. Per prendere un libro Bunny deve spostare tutti i libri sopra di esso. Bunny è molto pigro, e quindi vuole riuscire a prendere **due qualsiasi** dei libri a cui è interessato spostando il minor numero di libri possibile!

Qual è il minimo numero di libri che Bunny deve spostare, **contando anche i libri di giardinaggio che prende**?

Domanda 1

☐ (A) 2
☐ (B) 4
☐ (C) 5
☐ (D) 6
☐ (E) 7

☒ Lascia in bianco

Termina e controlla
 Nessuna risposta

Fig. 6. User interface for question 1 (multiple-choice, logic thinking).

Question 4.1. *Bunny found these five slips of paper with numbers written on them: [. . .] Bunny wants to know which numbers can obtain by aligning the slips vertically and reading a column. For instance, aligning slips as in the picture, he can obtain 14518: [. . .] What is the largest number he can obtain in that way?*

Question 4.2. *Actually, Bunny would also like to figure out what is the biggest number he can get if he can change the order of the slips. For example, exchanging the first sheet with the last one, the order of the sheets would become: [. . .] What is the largest number he can obtain in that way? (see Fig. 7)*

Question 4.2 is the one with the most percentage of incorrect answers (see Fig. 2). Even though lower secondary school students performed better on the two algorithmic questions shared with primary school students, they still had issues on the more complex follow-ups of them, showing that algorithmic thinking is indeed a skill that needs to be further encouraged and developed also in their age group.

11,581 students took part in the first phase: 4,274 from primary school and 7,307 from lower secondary school. The school with the highest number of participants was the “I. C. L. Da Vinci/G. Carducci” school in Palermo, with 633 students. 86 students obtained a full score: 38 from primary school and 48 from lower secondary school. The school with the most full scores was the “I. C. Torgiano-Bettona” primary school, with 13 full scores. The mean score was 19.5 and the median score was 20, the same for both primary and lower secondary schools. The reported satisfaction was high: the students enjoyed learning and showing off their potential, through a test deemed adequate in all its sections, and effective administration tools despite some minor technical difficulties.

4. Second Phase

Students who achieved sufficient results in the first phase were invited to participate in a second phase dedicated to coding which involved carrying out the questions using the computer. Also in this case, the test was prepared at a national level by the technical-didactic operational unit of the Italian Informatics Olympics committee. The competition consisted in solving algorithmic problems by writing computer programs. The programming language used was a choice of Python, Scratch, or Pseudo-code based on suggestions from the school teachers. The test consisted in three tasks to be completed in two hours, different for the two school levels but with an overlap: primary schools had tasks *mele*, *dadi*, *monologo*; while lower secondary schools had tasks *dadi*, *monologo*, *soldatini*.³ The students who obtained the best results in this second phase were invited to carry out the tests of the third phase.

1,320 students took part in the second phase: 295 from primary school and 1,025 from lower secondary school. Unfortunately, the test turned out to be very difficult and only half of these managed to score points: 142 from primary school and 442 from

³ An interactive training version of the exercises is available at: <https://demo.fibonacci.olinfo.it>

lower secondary school. There was only one full score for primary school, and six full scores for secondary school. 25 primary school students with a score of at least 55 points, and 57 lower secondary school students with a score of at least 100 points were selected for the national final. The feedback gathered from teacher was varied, but leaning on the negative side overall, as the test was discouraging most students. Particularly negative was the interaction with the test system for Scratch: as we weren't able to integrate Scratch within our platform, students were required to download and re-upload multiple files between two sites, resulting in a cumbersome and confusing interaction.

5. Third Phase

67 students participated in the third and final phase: 21 from 7 primary schools and 46 from 18 lower secondary schools. Of these, 18 primary school students and 36 secondary school students managed to get points. Among these, 32 medals were awarded, following the assignments used in other scientific competitions: 14 bronzes, 12 silvers, and 6 golds. The test consisted in four programming tasks to be solved in three hours. Three tasks were borrowed from the regional selections of the Italian Informatics Olympiads (*rettangolo*, *newlines* and *muro*), while the fourth easier task was specific to the competition (*formiche*).⁴ Every task was solved by at least a contestant, but no contestant solved every task, so that the maximum score was of 118 points out of 200. This was a satisfactory result for the contest itself, but quite far from the level that higher secondary school students achieved on the common tasks. The students selected for the national Italian Informatics Olympiads all scored at least 110 points on the three common tasks, with all of them fully solving the easier of the three tasks (*rettangolo*). As the additional *formiche* task was even easier, this projects their likely cutoff score to be at about 160 points out of 200. Since even the best scoring student from lower secondary school was quite far from this cutoff, we decided to not invite any student from this competition to join the Italian Informatics Olympiads, as we feared that would not be a constructive experience for them.

The feedback for this phase was positive overall: even though the test had the same format of the second phase, which was not well received, the more selected pool of students was able to handle the system effectively. This is not surprising as only the students performing well on the second phase were selected for the third, which are students that were already able to work successfully with the competition format previously. On the other hand, it is an indication that the format and level of the third phase was indeed appropriate for a national-level final competition for lower secondary schools, as sufficiently many students nation-wide were able to score well. Unfortunately, the same can not be said for elementary schools: only one student scored well (obtaining a lower gold medal with 104 points out of 200), while the other 17 participating students scored at most 11 points out of 200 (which were not enough for any medal).

⁴ An interactive training version of the exercises is available at: <https://demo.fibonacci.olinfo.it>

6. Lessons Learned

Based on the feedback and results gathered, we concluded that the competition should live for the following years, but with several necessary changes. As the first phase was the most successful, we plan to leave it mostly untouched, only reducing slightly the weight of logical quizzes in favor of algorithmic and program interpretation. We also plan to propose the programs to be interpreted in a block-like format, in order to be more preparatory for the following phases; and making our first phase more of a “further step” after Bebras. In this way, we plan to perform slightly more selection before the second round, to ensure that most of the students proceeding in the competition have the skills needed to take profit from it.

As the second phase was the most unsuccessful, we decided to completely rethink it. Instead of having it with an identical format as the *third* phase (but with easier tasks), we plan to have it with a similar format as the *first* phase, having it hosted on the same quiz-based web platform, with the goal of making the second phase somewhat more of an intermediate step between the first and the third. The second phase will differ from the first in two main aspects:

- The removal of the section on logical quizzes, thus only focusing on algorithmic thinking and program reading.
- The algorithmic questions will be solvable by writing a program in Blockly⁵ computing their answers.

More precisely, each algorithmic question will feature a Blockly editor integrated with the website, and multiple answer boxes for inputs of increasing complexity (inspired by the two-step questions asked in the first phase for lower secondary schools). By composing a simple program solving the question and pressing a “run” button, it will be possible to automatically fill in all answers and see whether they are correct. Only the first input will be small enough to be solved by hand, so that being able to correctly write a block-based solution should result in an higher score.

Finally, as the third phase was successful for lower secondary schools, we plan to leave it mostly untouched, but propose it only to lower secondary school students. Few selected primary school students may be invited as well if they score well enough, with their teacher’s consent, but by being an exception rather than the rule we hope that having very few primary school students at the third phase in this way should not detract from the sentiment of the (many) primary school students that will not be selected. In fact, we feel that we cannot propose a full-fledged programming contest to primary school students in Italy at the time of this writing (except for very few exceptional students).

⁵ The choice of relying on Blockly instead of the more well-known similar platform Scratch is due to its easier integration with our website. Blockly can be tried online at: <https://blockly.games>

- Kanemune, S., Shirai, S., Tani, S. (2017). Informatics and programming education at primary and secondary schools in Japan. *Olympiads in Informatics*, 11, 143–150.
- Kiryukhin, V.M., Kinder, M.I., Cvetkova, M.S., Borisov, N.A., *et al.* (2022). Methods of tracks for training juniors in Olympiad Informatics: The ISIJ experience. *Olympiads of Informatics*, 16, 75–87.
- Pluhár, Z. (2021). Extending computational thinking activities. *Olympiads in Informatics* 15, 83–89.
- Souza, I.M.L., Andrade, W.L., Sampaio, L.M.R., Souto O Araujo, A.L. (2018). A Systematic Review on the use of LEGO® Robotics in Education. In: *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE. 1–9.
- Vegt, W. van der. (2016). Bridging the gap between Bebras and Olympiad: Experiences from the Netherlands. *Olympiads in Informatics*, 10, 223–230.



G. Audrito is involved in the training of the Italian team for the IOI since 2006, and since 2013 is the team leader of the Italian team. Since 2014 he has been coordinating the scientific preparation of the OIS and of the first edition of the IIOT. He got a Ph.D. in Mathematics in the University of Turin, and currently works as a Junior Lecturer in the University of Turin.



M. Ciobanu is involved in Italian Olympiads in Informatics since 2009. She got a Ph.D in Computer Science in the University of Salerno, and currently she is a teacher in a high school.



L. Laura is currently the president of the organizing committee of the Italian Olympiads in Informatics that he joined in 2012; previously, since 2007, he was involved in the training of the Italian team for the IOI. He is Associate Professor of Theoretical Computer Science in Uninettuno university.

The Introduction of Micro:bit in Elementary School, from Unplugged Activity to Programs

Bence GAÁL

*ELTE Faculty of Informatics – Department of Media and Educational Informatics, Budapest
e-mail: gaalbence@inf.elte.hu*

Abstract. In this article we would like to present a good practice and its results, in which the main role is given to understanding the micro:bit device and its programming. Starting from an unplugged activity that helps understand the device's functioning and basic concepts, the workshop leads fourth-grade students to create simple, yet impressive programs written in the device's block-based language. The research was focused on investigating the success of the unplugged activity developed specifically for micro:bit, as well as on assessing the motivation of the students and the level of knowledge that could be transferred to the students through this method. It was also important that, in addition to the results, the practice material, with a detailed description, should be available to other teachers who would like to introduce micro:bit to their students in this way.

Keywords: unplugged, robotics, micro:bit, programming, STEM, elementary school.

1. Introduction

In Hungary, before 2020, informatics was only present as an optional subject in school education for students in grades 3–4. However, in most schools, informatics does not appear among the optional subjects, which is probably due to the fact that career abandonment is highest among informatics and engineering teachers, and the ageing teaching population is more difficult to keep up with the development of informatics (Gaál, 2020). The situation will definitely be improved by the new National Core Curriculum, where digital culture is already present as a compulsory subject in grades 3–4 (Education Authority, 2020a).

The students of fourth grade participating in the experiment first encountered informatics in classroom conditions during the unplugged session, as they were still under the old curriculum at the time of writing the article. The topic focused on programming the micro:bit and getting to know it, but we felt it was important to introduce the topic in an unplugged way. The Computer Science (CS) unplugged methods are extremely effective in that students see another, interesting side of informatics, namely what hap-

pens inside the computer (Rivka *et al.*, 2012). It is important that their first experience be appropriate for their age and that they encounter different logical and programming problems through activities where they do not have to write program code to solve them (CSERG, 2021). Therefore, in the first half of the session, we simulated the running of a micro:bit program in an unplugged activity, through which the children were introduced to the device and gained an understanding of how it works, as well as some basic programming concepts.

The second half of the session focused on creating short (robotics-related) programs. We tried to present programs that were sufficiently attention-grabbing and enjoyable for the students. Before the method is implemented, we also set up several hypotheses, which were followed by a questionnaire for both the unplugged activity and the programming part, which we will elaborate on in the second half of the article.

[Hypothesis 1]: By the unplugged activity, students will be able to distinguish the different elements of the micro:bit, and be familiar with some basic concepts that only older age groups know.

[Hypothesis 2]: Even just one of these robotics lessons can spark children's interest and motivation to work with robotics, in their informatics and science lessons and even in their free time.

For other future research, it was important that students were also open to using micro:bit in other subjects.

2. Presentation of the Lesson Material

We used the new V2 version of the micro:bit for the sessions (Fig. 1). The device has undergone a lot of innovation and many new features that are interesting for children have been integrated. In addition to the increased memory and more modern processor, the device also received a capacitive touch sensor, speaker, and built-in microphone. We also tried out the latter function with the fourth-grade students. It is important that during



Fig. 1. The front and back of the micro:bit V2
(Source: <https://microbit.org/new-microbit/>).

the development of the device, one of the key topics of the 21st century, environmental awareness, was also taken into account and the device became more energy efficient and already has a sleep state function (MEF, 2021; Abonyi-Tóth, 2021).

The implementation of the session took place in two parts, which took four lessons, but this can be greatly influenced by the group dynamics and the individual needs of the students.

The first half of the session was devoted to testing and verifying the unplugged method developed during my studies. This was preceded by a detailed presentation of the device so that students have an idea of what we are going to work with. The essence of the unplugged activity is to familiarize students with the internal functioning of the micro:bit, to show them playfully what operations the processor has to perform and how the device communicates with the user. This also clarifies the concept of input and output peripherals. The first step in implementation was to select from among the students those who played the roles of different parts of the micro:bit (Fig. 2). The students were given the role of certain parts of the micro:bit, for example, the “Managers of the screen pixels” were given the task of plotting the output on the board as if it were the LED matrix of the micorbit, and the students in the role of “buttons” were given the task of indicating when they were pressed. Those who did not want to take on these roles were given the programmer role. They were the ones who created the program commands using statements cut out of paper and passed them to the CPU. The preparatory requirement for the session was to print out the blocks according to the task. And through the unplugged activity, students simulated the entire programming process, from coding to switching on the LEDs, by personifying the processes of the device themselves.

After the preparation, students in role of “programmers” independently try to create the program code from the blocks. In our case, an animation about a waving robot (Abonyi-Tóth, 2018) was the first program to be implemented. The code causes the image of a robot to wave with its left or right arm by pressing the appropriate button.

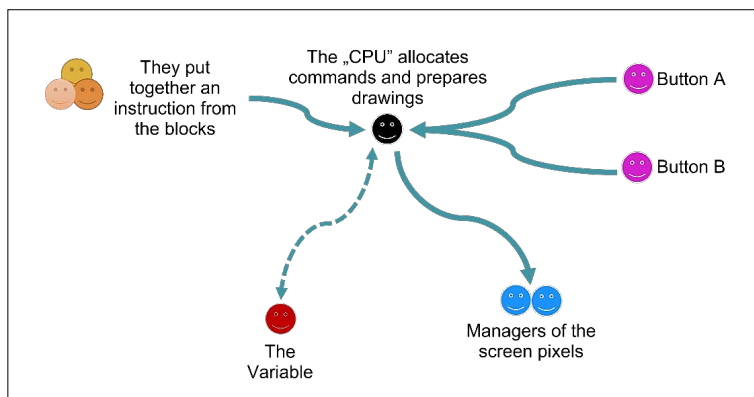


Fig. 2. How to implement the method
(Note: the use of variables was not discussed during this exercise).

The instructions are received by the student in the “processor” role and based on these it notifies the students in role of another necessary micro:bit parts and assigns the tasks, e.g.: *the button sends a notification if it is pressed*. The display can be implemented at the board. Here we can assign more students for the role of “managers of the screen pixels” to speed up the process, as each movement has to be drawn out. With this role, we can ensure that everyone is part of the activities even in larger groups. We can further increase the number of roles, if necessary, for example, we can also distribute messenger roles. After we played a few steps, discuss with students what happened during the process. For a better understanding, it is recommended to project the schematic diagram that students can see the direction of the arrows and understand why the directions are important (Fig. 2).

The questionnaire examining the success of the unplugged activity was completed after this part of the session. (see Chapter 3) The second part of the session involved implementing the previously created program in a digital environment as well, using the makecode interface. The length of solving the task was greatly increased by the fact that they were fourth-grade students, for whom this was the first informatics lesson. And for us to get the micro:bits working, it is essential to know file and folder operations. During the session, perhaps this was the most difficult part of the material for the students. Once the animation was already waving with one hand, it was up to the students to get the other hand working as well (Fig. 3), and to create individual animations with both buttons pressed. Implementing this did not cause any problems after the unplugged activity and the implementation of the first version of the code.

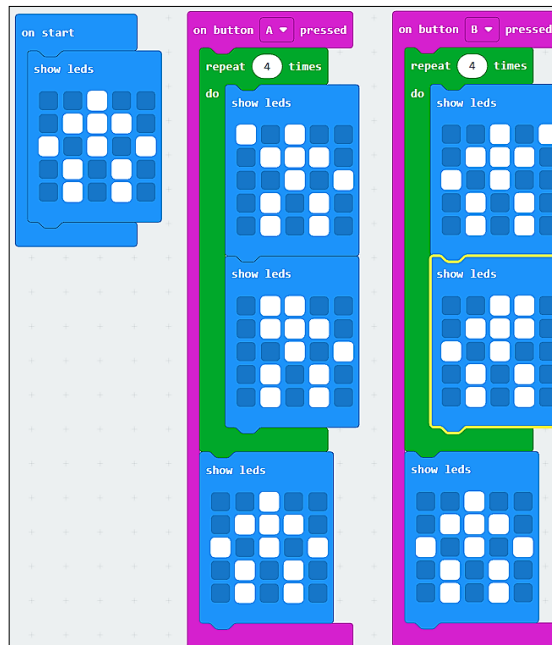


Fig. 3. Example code of the waving robot (Abonyi-Tóth 2018).

The second task we prepared with the students was a graph that changes with sound. Here, the students could choose their favorite music, which, when played on a smartphone, provided a perfect effect for the spectacular presentation of the program code. Here the emphasis was more on the spectacle than on the difficulty of programming, this was sort of a relief for them, as they learned a lot of new things in a very tight pace. At the end of the session, the stem:bit accessory package developed for micro:bit was also presented and students could also take a look at an obstacle-avoiding crawler robot.

This part of the session also ended with a questionnaire. (see Chapter 4) This questionnaire focused on the motivation of using micro:bits and the feelings of the students related to the lessons.

3. Testing the Success of the Unplugged Activity

The questionnaire related to the activity contained four multiple-choice test questions and two open-ended questions. The test was completed by all of the participant students ($N = 18$). The questions cover computer knowledge, of which only the concept of a “program” appears in grade 4. The other concepts (loop, sensor, peripheral, processor) are only included in the digital culture curriculum for grades 5–6 (Education Authority, 2020b/c). The goal was for students to have some clarity about these concepts, even after only one lesson. In the following, we will go through the results of the tasks one by one.

Question 1. What is the processor for?

Out of the 18 respondents, only one person gave a wrong answer, marking the third option. The other respondents correctly marked the second option (Fig. 4). It can be concluded that the group understood the basic role of the processor.

Question 2. What the sensors do?

In this case, the understanding was made more difficult by the fact that only the concept was mentioned verbally, its presentation did not take place within the frame-

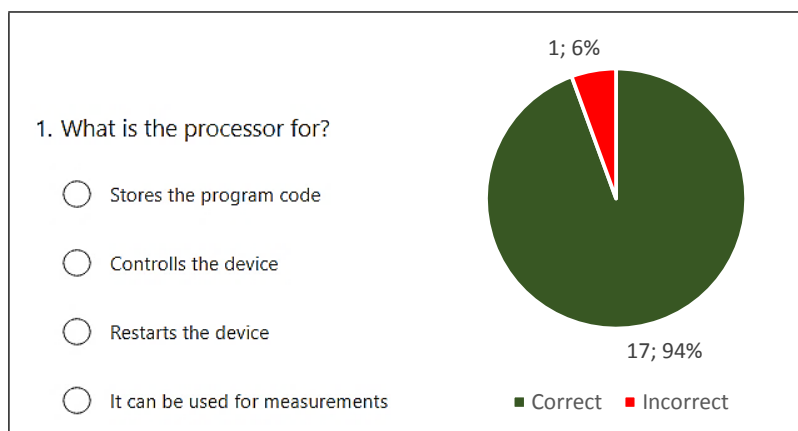


Fig. 4. Question 1 and the distribution of responses.

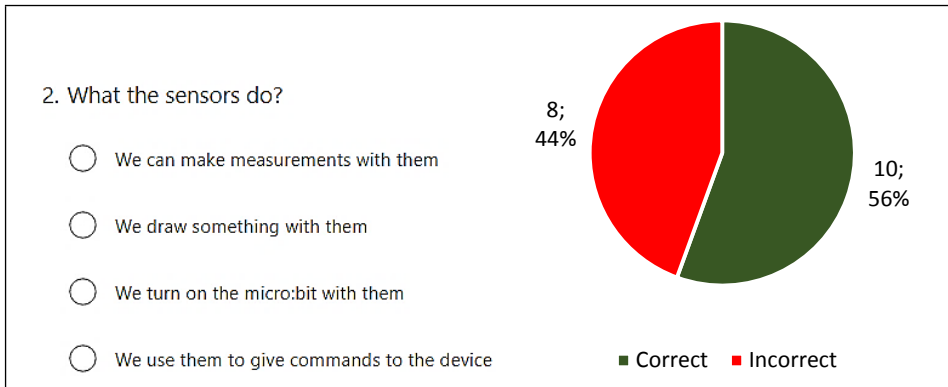


Fig. 5. Question 2 and the distribution of responses.

work of the unplugged activity. Since learning did not take place through experience, it was expected that we would get a weaker result. Accordingly, 8 students gave incorrect answers out of the group. Of these, 5 students marked the second, 2 the fourth and 1 the third option. 10 students correctly gave the first option as an answer (Fig. 5).

Question 3. What peripherals are the buttons?

Question 4. What peripheral is the display?

The next answers are closely related to each other, as they had to decide on the different parts of the micro:bit (buttons, display) whether they are input or output peripherals. At these questions, convincing results were also achieved. Most of the children correctly distinguished between input and output peripherals (16 correct answers for buttons, 15 for display) that are on the device. Distinguishing this is still difficult not only in grade 4 but in grade 5 according to our experience. Two students gave wrong answer to both questions. It can also be stated here that as a result of the activity, they answered correctly in a large proportion to the question asked.

Question 5. What a computer program is?

The next question was an open-ended question. This is, according to the new curriculum, already some knowledge to be learned in grade 4, but as we wrote earlier, the students who participated here, did not have digital culture or informatics as a subject either (Education Authority, 2020b). Considering the age characteristics, we accepted all those answers as correct solutions that included the following expressions: *it contains instructions*, *it gives instructions to the computer*, or *instructions after each other*. The given definition during the activity was the following: a set of instructions given to a computer (Gregorics *et al.*, 2012).

Almost three-quarters of the group, 13 students, gave an appropriate answer to the question, so we think that the activity may be suitable for them to learn what a program is (Fig. 6). In the case of wrong answers, 3 students also wrote that the program is the brain of the micro:bit. This answer may be interesting if we think about how the students thought. Perhaps they tried to solve the problem with something closer to them and drew a parallel with the relationship between human action and the brain, because our brain also gives instructions to our body.

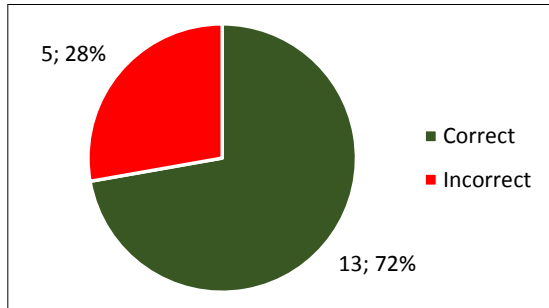


Fig. 6. Distribution of responses to Question 5.

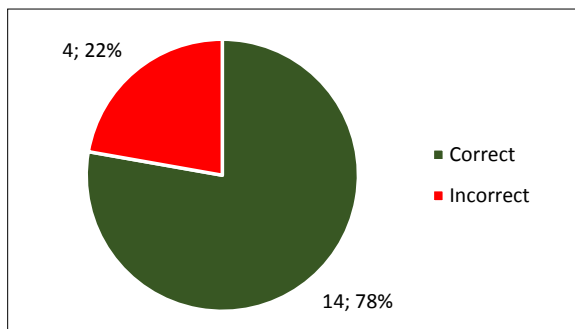


Fig. 7. Distribution of responses to Question 6.

Question 6. What does a loop do?

Similar to the previous question, this was an open-ended question too. The given definition was the following: the loop repeats the instructions it contains several times. (Gregorics *et al.*, 2012). This concept is not part of the grade 4 curriculum, and only appears in grade 5 (Education Authority, 2020c). As correct answers, we accepted the following: instruction repetition, repeating part, repeat something/process, repeat the thing several times.

The result here is also convincing. Out of the group, 14 students gave a correct answer to the question and only 4 students failed the task (Fig. 7). It can therefore be stated that the activity may also be suitable for illustrating the loop.

Success of the activity

The activity was successful according to expectations. It can be said that in every question, at least 70% of the students answered correctly from the questions asked, which we covered during the unplugged activity. Breaking down the results by individuals, everyone reached 50%, even students with weaker digital skills and children who had not used a computer at all before. They were able to distinguish and understand the functions of the different parts of the device, such as button, display, processor. The implementation of the activity can of course also work through another example program, where not only the loop but also the if-else statement can be taught to the students.

Based on the above and the results of the questionnaires, it can be stated that the [H1] hypothesis that By the unplugged activity, students will be able to distinguish the different elements of the micro:bit, and be familiar with some basic concepts that only older age groups know., has been fully confirmed.

4. Results of the Satisfaction

Within the framework of this questionnaire, we were mainly interested in how much the students enjoyed the activity and how the whole activity affected their motivation related to robotics. The response was on a Likert scale ranging from 1 to 5. The questionnaire was completed by 16 students. In the meantime, one student had to leave, and one student instead of filling out the questionnaire started to further develop the program of the lesson. In his case, the activity can definitely be considered successful. In the following, we will deal with the distribution of answers to each question.

Question 1. How interesting did you find the lesson?

For the first question, all the students rated the value five, so everyone found the lesson very interesting. Robotics could play a big role in this, as it is by its nature a subject in which children are more interested. It should also be noted that the teacher's presentation style also influenced the answers. An interesting approach without a committed, enthusiastic teacher is less likely to be successful. Nevertheless, the fact that the students all gave the highest marks shows that the method tested is capable of delivering an interesting lesson for all.

Question 2. How much fun did you have?

The second question was "How much fun did you have?" The average of the given values is 4.9375, as 15 of the students marked grade five and only 1 person ticked the answer option of four, rather yes. The fact that 15 students gave the highest rating of five suggests that a significant proportion of the class had a very enjoyable experience of the activity or lesson. This positive response can be attributed to a variety of factors, such as engaging content, interactive teaching methods or the general classroom atmosphere. The last factor is not significant in this case because the trainer and the group were unknown to each other. This implies that the content of the lesson largely determined the answers.

Question 3. How much would you like to use the device in your free time?

The last question was "How much would you like to use the device in your free time?" In this case, the opinion of students was somewhat divided. If we look at the average, the response value was 4.4375 and the standard deviation was 0.629. The distribution was as follows. 8 students would love to use the device, 7 would rather use it, and 1 person would not use it. Responses for leisure use are less consistent. In our opinion, this is not a problem, since the aim of using the tool is to make classroom activities interesting and to introduce programming effectively. Nevertheless, half of the students found the tool interesting enough to want to use it in their free time.

It may be worth exploring the background of the negative response to find out the personal preference of the student and the underlying content of the negative response.

Question 4. Would you like to use the tools in your informatics class next year? (Yes or No question)

All 16 students responded positively, confirming that robotics has a place in the new curriculum. Consequently, in the group's subsequent informatics lessons, we were able to put a strong emphasis on micro:Bit programming, which was mainly implemented in pairs.

Question 5. Would you like to use the tools in your science class next year? (Yes or No question)

The general unanimous positive answer to this question provided a good basis for further research, focusing on robotics-enhanced science education. The enthusiasm and openness of the students allows the benefits of robotics and micro:bit to be used in other subjects. This question and the previous one were intended to assess the group's openness towards micro:bit in order to determine whether further research with the group would be possible in the future.

Summary of the satisfaction questionnaire

The answers to the satisfaction questionnaire were overwhelmingly positive for all questions. Students enjoyed the lessons and found it interesting, and most of them would use the micro:bit at home in their free time. In addition, all the students were open to using the device in future computer science and science lessons, thus getting to know them better. It can therefore be clearly concluded that the [H2] hypothesis that even just one of these robotics lessons can spark children's interest and motivation to work with robotics, in their informatics and science lessons and even in their free time, has also been confirmed.

5. Conclusion

The integration of robotics into education is inevitable. Therefore, we would like to design an activity that gives students the opportunity to explore the world of robotics in a playful way. During the experimental session, the participating group 4, without any previous knowledge of informatics, was able to write programs on micro:bits and modify them individually during a longer session. The playful introduction of the topic, which introduced and personalised the micro:bit in an unplugged activity, played a major role in this. Its success lies in showing students, through an easy-to-understand role-play, the complex and abstract process of programming and the basic principles of micro:bit operation.

Based on the results reviewed in this paper, it can be concluded that the use of robotics and unplugged methods positively influences learners' understanding and helps learners to acquire knowledge beyond their age, as learning for them is playful and active, where the mechanism of playful learning is implemented.

The beneficial effects of robotics should also be highlighted, as even a single session greatly influences how students relate to the given subject. We encourage all colleagues to use these robots not only within the framework of digital culture, but also to integrate

them into natural sciences, as it seems that students would like to see them there as well. With the help of these tools, motivation and skills for STEM subjects can be easily developed, and they also provide a great opportunity to develop soft skills during education with the appropriate methodology, where group- and pair work and project-based education are emphasized. Through such processes, students will have a great opportunity to gain an experience that will benefit them in many situations later in life.

References

- Abonyi-Tóth, A. (2018). *Programozunk mciro:Biteket!* ELTE Informatikai Kar 1117 Budapest, Pázmány Péter sétány 1/C. 23–25. Retrieved May 30, 2023, from <http://microbit.inf.elte.hu/wp-content/uploads/2018/05/Programozunk-microbiteket-2018.pdf>
- Abonyi-Tóth, A. (2021). *Micro:bit V2 – többet, jobban, gyorsabban, továbbra is olcsón...* Retrieved May 30, 2023, from <http://microbit.inf.elte.hu/2020/11/26/microbit-v2-tobbet-jobban-gyorsabban-tovabbra-is-olcson/>
- Computer Science Education Research Group – CSERG (2021). <https://csunplugged.org/en/about/>
- Education Authority (2020). *Digitális kultúra – Kerettanterv az általános iskola 1–4. évfolyama számára.* Retrieved May 30, 2023, from https://www.oktatas.hu/pub_bin/dload/kozoktat/kerettanterv/Digitalis_kultura_A.docx
- Education Authority (2020). *Digitális kultúra – Kerettanterv az általános iskola 5–8. évfolyama számára.* Retrieved May 30, 2023, from: https://www.oktatas.hu/pub_bin/dload/kozoktat/kerettanterv/Digitalis_kultura_F.docx
- Education Authority (2020). *Nemzeti Alaptanterv.* Retrieved May 30, 2023, from <https://magyarkozlony.hu/dokumentumok/3288b6548a740b9c8daf918a399a0bed1985db0f/letoltes>
- Gaál, B. (2019). *A robotika témakör integrálásának lehetőségei a természettudományos tan-tárgyak oktatásában*, InfoDidact 2019, 59–72.
- Horváth, G., Gregorics, T., Heizlerné Bakonyi, V., Menyhárt, L., Pap, G. S., Papp-Varga, Z., Szlávi, P., Zsakó, L. (2012). *Programozási alap-ismeretek. Eötvös Loránd Tudományegyetem, Informatikai Kar.* Retrieved May 30, 2023, from http://progalap.elte.hu/downloads/seged/eTananyag/lecke6_lap1.html#hiv4
- Micro:bit Educational Foundation(MEF). Meet the new BBC micro:bit. Retrieved May 30, 2023, from <https://microbit.org/new-microbit/>
- Taub, R., Armoni, M., Ben-Ari, M. (2012). CS Unplugged and Middle-School Students' Views, Attitudes, and Intentions Regarding CS. *ACM Transactions on Computing Education*, 1–29.



B. Gaál – PhD student and assistant lecturer at ELTE Institute of Computer Science – Department of Media & Educational Technology. He is a member of the Hungarian Bebras Organization Team. Graduated with an MSc degree in teaching Geography and teaching Informatics. His main research areas are the integrability of robotics in the natural sciences and the usage of micro:bit in education.

Online Robotics Activities During the Pandemic Period – Challenges and Experiences

Bence GAÁL

ELTE Faculty of Informatics – Department of Media and Educational Informatics, Budapest
e-mail: gaalbence@inf.elte.hu

Abstract. The situation caused by the Covid-19 made it impossible to keep our traditional in-person summer camps and activities, so our university decided to hold the sessions online. Thus, enthusiastic students aged 13–16 were able to participate in robotics sessions that were fully implemented in the online space. In the article, we want to present the experiences and challenges of these sessions, as well as the way of implementation and the range of tools needed for it.

Keywords: robotics, Covid-19, distance learning, IT education, learning activities.

1. Introduction

The robotics workshops that are the subject of our article were organized by ELTE T@T Kuckó in the summer of 2020 in three different weeks. A different group took part each week. The content of the curriculum was the same each week. The three-week split was necessary to make the groups easier to manage and to allow more time for individual assistance. Since the target group was 13–16 years old, we did not want to implement coding in a block-based programming environment. The workshops were primarily designed for students who had not programmed in python and/or had not yet encountered the micro:bit. The aim of the workshop was therefore to learn the basics of programming in addition to getting to know the device. During the three weeks, each workshop started with the maximum number of students (14 students) and there was no dropout.

When choosing the programming language, it was important to use a language that is ideal for beginners. The advantages of python include its simplicity, security, and support for object orientation. (John, 1999) Research also shows that students found python more fun and easier to learn than the languages they had used before. However, it is important to note that there are also disadvantages to using the language. Since it is a scripting language, performance loss occurs when running longer programs. Another disadvantage is the lack of information encryption and dynamic type assignments (assigning multiple types to a variable) (Grandell *et al.*, 2006). From an educational point of view, however, it is advantageous that language requires the user to create code in a

structured way because in python, indentations will indicate the beginning and end of code snippets. In other languages, readability is left to the programmer, while here the language requires us to code in a readable and “good looking” way (Donaldson, 2003). This helps students internalize this type of coding, which can be perfectly capitalized on in other languages as well.

When choosing micro:bit, we considered two main criteria. One was its excellent usability in education and the other was the price of the devices. Participants had to purchase their own devices to participate in the workshops. In addition to purchasing the device, no other costs were incurred by participants as the workshops were free of charge. Thanks to its versatility, it can be suitable for children with different interests and can be integrated into several areas of disciplines and programs related to the topic can be created (Abonyi-Tóth, 2018; Gaál, 2019). In addition, we talk about a compact device that can be easily extended even for beginner users.

2. The Curriculum of the Workshop

The main goal was to have enough knowledge and understanding of the device to implement a game on micro:bit by the end of the week. The main elements of the workshop’s curriculum were mainly based on the following workshop resources:

- Programming micro:bits in python – Norbert Szűcs¹.
- Programming micro:bits – Andor Abonyi-Tóth².

The curricula mentioned above must be adapted to the duration of the workshop, and the number of sessions, which took place in 5x2 hour intervals. Due to distance learning, we completely omitted the presentation of the device’s radio functions during the sessions, as this would have required more devices for the students.

Below we briefly explain the content of each session. Learning materials supported by full lesson plans can be found in the resources above. We tried to combine the classes with continuous independent assignments, which were submitted online. Do not forget to use these materials as a kind of outline and always try to customize the course of the lesson to the participants of the given course.

Session 1:

The essential part of the session was preceded by a so-called installation part. This is because the interface, which we will detail later, will only recognize devices with the latest firmware version, and only then will functions be available that greatly speed up workflows. This can be easily achieved by visiting the following website: <https://microbit.org/get-started/user-guide/firmware/>.

This was followed by the preparation of our first program, which was nothing more than the announcement of “Hello World!”. The rest of the lesson was spent drawing

¹ http://microbit.inf.elte.hu/wp-content/uploads/2019/10/microbit_python_szakkor_szucsnorbert.zip

² <http://microbit.inf.elte.hu/wp-content/uploads/2018/05/Programozzunk-microbiteket-2018.pdf>

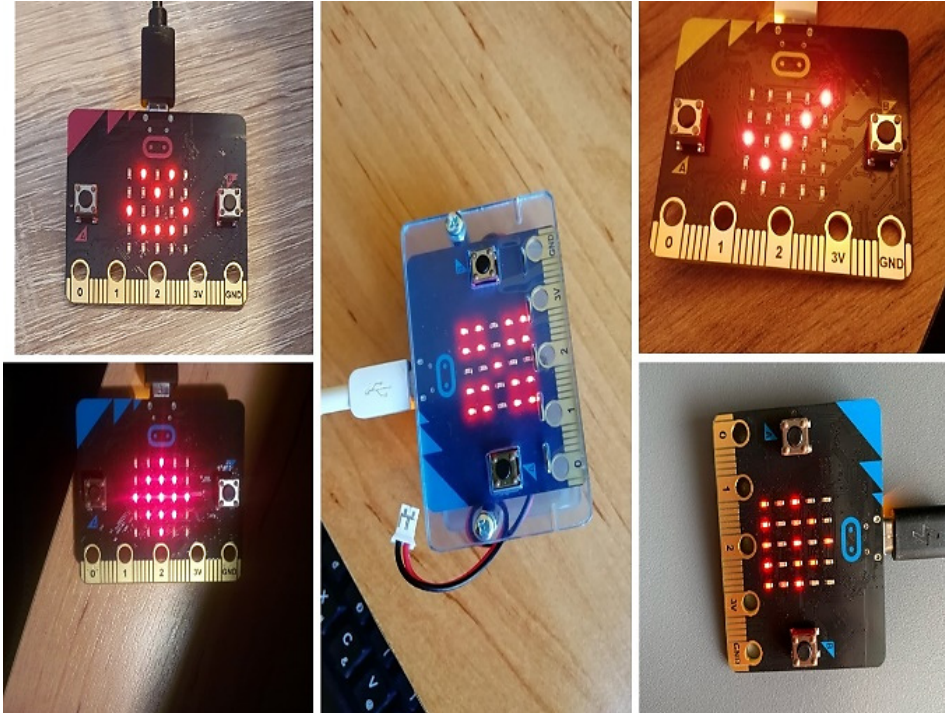


Fig. 1. Some results of students work on the first day.

shapes and animations on the micro:bit LED display. As an independent task, it was left to the children to take their own pictures (Fig. 1).

Session 2:

We also started the second session focusing on the display of the microbit. However, here we no longer displayed coherent images, but addressed the points of the LED matrix separately. Here random numbers and their role in IT came up. The attention of children can be captured very well by using their favorite computer games as examples and introducing new knowledge through them, so the presentation of random numbers is presented through the loot mechanism known from the games.

In the second half of the lesson, we implemented the animation of the bunk house available from the block curriculum in python language and the students also had time to implement a manipulated dice roll. It can be concluded that a playful and interesting approach to random numbers greatly facilitates the acquisition of knowledge by children and their ability to apply it.

This is where the basics of event-driven programming through the use of buttons were presented.

Session 3:

In this session, sensors already played the main role. After creating the compass, we created a jump counter application in python.

With the students, we reviewed other ways to give instructions to the micro:bit besides button presses, and then we created our own magic 8 ball³ program, where the data structure and operations of the list were already needed. The essential part of the implementation was the independent task of the students, after they became familiar with the commands for the new operations.

Session 4:

In the fourth session, we solved “do it yourself”-type tasks. This occasion was very popular among children during all three weeks. Here the following accessories were also needed:

- 4 crocodile clips with cable/lots of tinfoil.
- 2 pieces of nails.
- Pot and earth, which was preferably dry.
- 1 banana.
- 1 orange.

In the first half of the lesson, we made a fruit piano (Fig. 2), where we extended the micro:bit circuit with the help of ourselves and the fruits, thus playing sounds with the device. After making the piano, the students were given the task of making a siren.

In the second part of the session, we created a moisture meter (Fig. 3) that can be used in a smart home project, which monitors soil moisture in real time by monitoring soil resistance. Here an elaborate, micro:bit controlled irrigation system was presented, and the construction of smart homes was also discussed.

It was this task that really left its mark on the children, and taking sustainability and energy saving into account, many people thought about the task further and started to implement their own project.

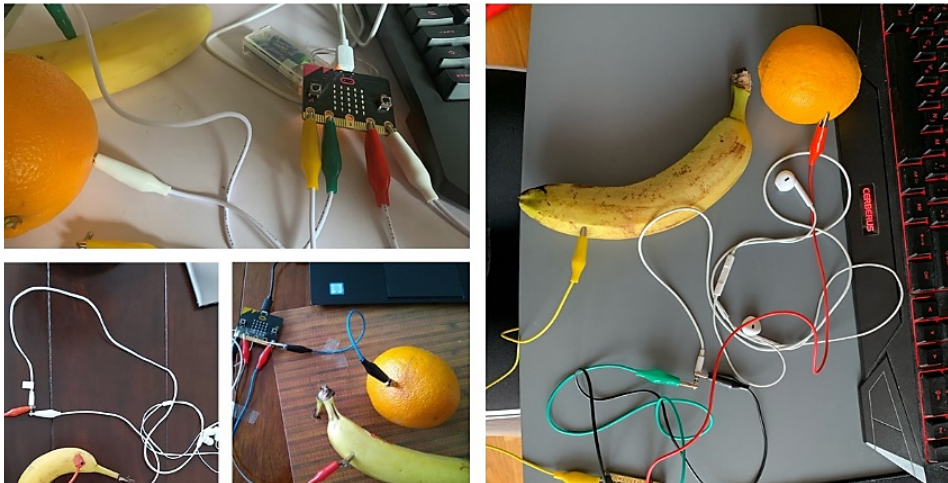


Fig. 2. Fruit pianos made by students.

³ In fairy tales and films, the magic 8 ball has repeatedly starred, which randomly answers our questions.

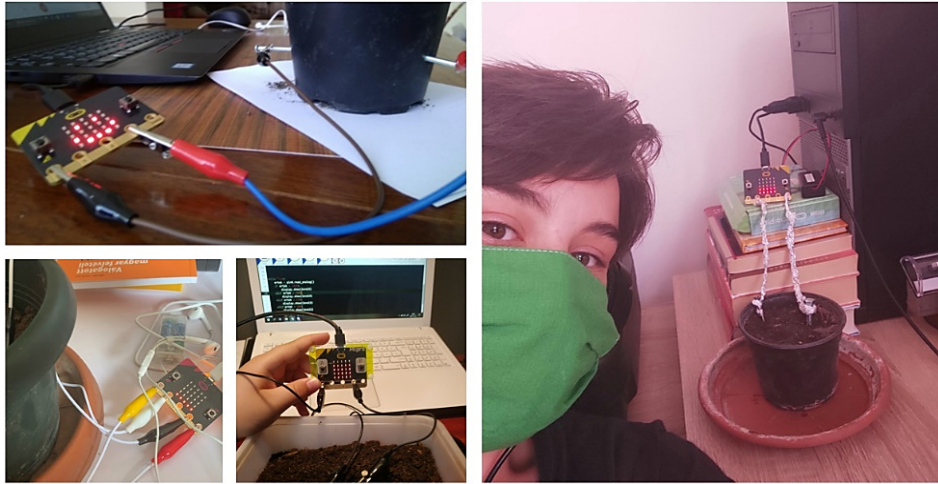


Fig. 3. The implemented moisture meters.

Session 5:

On the last day, the most awaited activity for the students took place. Here we implemented Flappy Bird on micro:bits. During the process of creating the game, we built heavily on what we have learned so far, and almost every single element we learned has been put into use. The students can now use what they had learned to create a complex program and receive feedback on how well they had mastered this knowledge.

During the creation of the game, object-oriented programming appeared as a new knowledge, and the participants gained insight into the different development phases. The programming process was divided into parts, and some sub-problems had to be solved by the students in an independent form of work.

3. Software Requirements of the Workshop, Presentation of the Virtual Environment

During the workshop, we used several software to create the most ideal environment for online education for students. We tried to choose programs that are suitable for organization and task assignment, holding online classes, visualization and of course coding and its supervision. Below we will review these platforms.

The virtual classroom and sessions

For this part of the implementation, we used Google services. On the Google Classroom interface it is possible to organize groups, assign assignments, and share information with students. It is perfectly usable for traditional school activities as well, as Google offers us a complete LMS.

In the “Classroom”, we can enter the schedule into the group calendar, which can be associated with a conference call using the Google Meet application. “Meet” also has all

the features needed to conduct online meetings. Children can easily start screen sharing, so the teacher can provide personalized help to them.

Several factors influenced the selection criteria. We needed a software package that not only allowed for classroom organization but also for holding online classes. It was also important to be able to easily integrate different file types into the site, as tasks had to be assigned and students had to upload pictures and videos. We needed a system that most students were familiar with and could easily handle and access. The Google software package fully met these criteria and only required an email address to access it.

If students under the age of 16 want to use Google accounts, logging in with their parent's email address can be a solution in non-institutional cases. However, institutions have the option of applying for the G-Suite system, which allows for a uniform email address to be given to students and these accounts can also be supervised by those with system administrator privileges. It is also worth noting that Google's software package is perfectly usable on multiple platforms, so students do not necessarily need a computer.

The virtual board

We would like to highlight a possibility for replacing board drawings in the case of online classes. We believe that board drawing is an indispensable pedagogical visualization tool in teaching programming, which helps students better understand processes and representations at an abstract level.

In our case, we used one of Google's applications for this, namely "Jamboard". In addition, we had a tablet with a pen. This can be replaced with a simpler drawing tablet or, if there is no tablet in our toolkit, we can draw with a mouse. Jamboard is a good choice because our virtual notebook can be shared with others and thus students can receive notes and create products together.

During the workshop, we opened the Jamboard application on the teacher's computer and participants could see what we were drawing on the tablet on the shared screen. This solution, where they do not open another application but only see the result projected, is a better solution so as not to overload their devices too much, as participants are not equipped with uniform strength hardware.

The online programming interface

We haven't talked about the programming interface yet. Online programming and collaborative document editing is not a new thing. Instead of well-established sites like repl.it, we chose a site that is still in beta but was specifically designed for the micro:bit. This interface is the micro:bit classroom.

On the site, we can program not only in python but also in a block-based environment. After selecting the language, the website generates an entry password consisting of icons and a pin code. This ensures that unauthorized persons do not enter the group.

In this article, we will focus on presenting the coding part of the website (Fig. 4) and explore its possibilities. The interface is perfectly usable in the field of online education. Its outstanding usability lies in providing solutions that make it easier for students to connect between the device and the computer and provide good opportunities for teachers to bridge distances. We will detail these properties below.

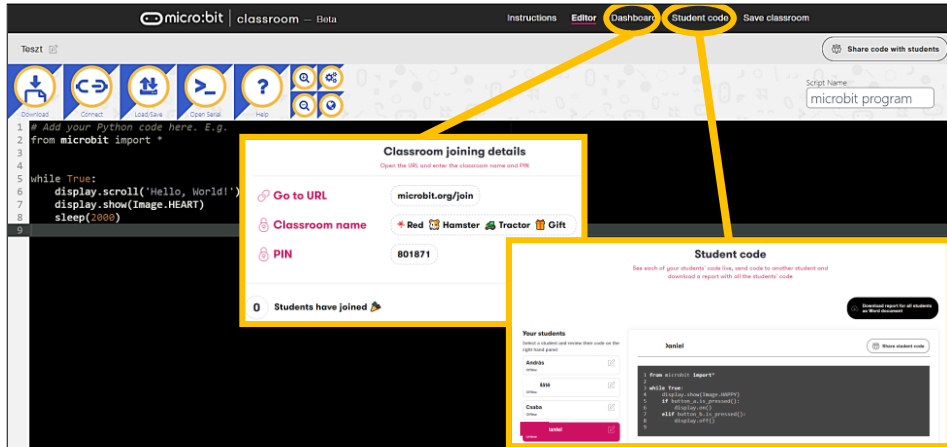


Fig. 4. Old interface of micro:bit classroom.

At the time of the workshops, the code editor was nothing more than a classic python IDLE. This was perhaps the only drawback of the environment that it does not contain any kind of verification system, as in Visual Studio, for example, or on the repl.it page mentioned earlier. Since then, however, an interface has been created that mixes block and text programming, and students can insert ready-made python code snippets in drag and drop principle. In addition, the editor provides tips related to various commands and debugging has been implemented. (Fig. 5)

We can also download our finished code, but if we have connected a micro:bit, we can skip all file operations. With a simple click, we can load our code directly onto the connected device. The connection of the device should be implemented as described earlier. As teachers, we can view the code of all students in the classroom and with one

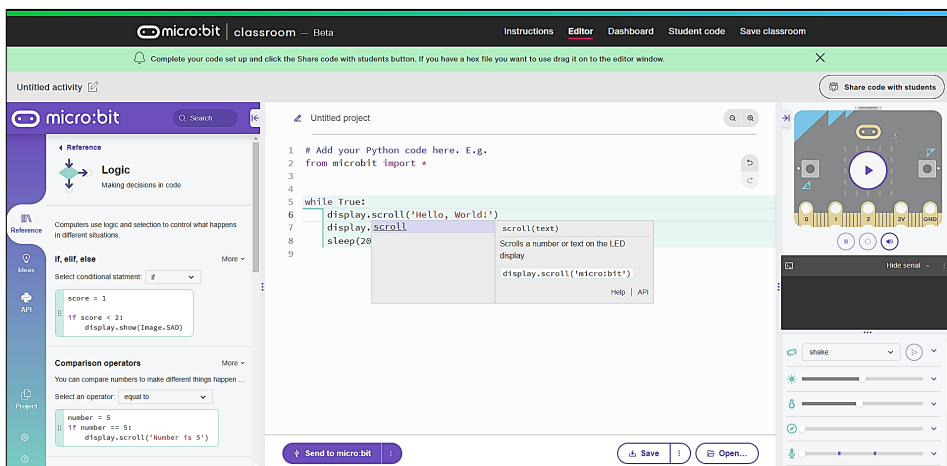


Fig. 5. The current version of micro:bit classroom interface.

click we can choose which student's code we want to see and easily help them. There is no need for screen sharing or other settings, and it is also a great advantage from a teacher's point of view that we can send the code written by us to all or selected students with one click. Students will be notified of this and can accept our code or continue working with their own. It is extremely useful for issuing framework programmes. In addition, we have the possibility to create a document of all codes and save it. This document will contain the codes of each student and structure them for us.

The interface was created specifically for micro:bit, and considering all the positive features mentioned above, it can be concluded that we can see the realization of an almost perfect concept on the micro:bit classroom site. Compared to the beta version stage, there were very few bugs and it has now taken the place of the text editor, a semi-block interface that is much more friendly to students.

4. Final Thoughts

Even within the framework of digital education, robotics can motivate children and stimulate their interest in programming. The group can be considered completely successful, as anyone had the opportunity not to come to the next event, but fortunately no one took advantage of this, and all students participate in every event. The participants had a good time during the workshop and perhaps despite the situation at that time, they managed to bond a little. There were several examples of students reaching out to each other during the afternoon as well, about learning or playing together.

Approaching it from the side of the device, it is important to note that we could make better use of the functions of the device if we are attending a classic classroom class. Nevertheless, we should not discourage holding online workshops either, as this way students who would almost certainly not have been able to attend robotics classes could not have been introduced to robotics by overcoming the distance, as they live hundreds of kilometers away from Budapest. We believe that with the help of online workshops we can reach more children, thus transferring the knowledge from which they can discover the world of programming and the functions of the given device on their own.

References

- Abonyi-Tóth, A. (2018). A micro:Bitek felhasználási lehetőségei az oktatásban. InfoDidact 2018. Retrieved May 30, 2023, from <https://people.inf.elte.hu/szlavi/InfoDidact18/Manuscripts/ATA.pdf>
- Gaál, B. (2019). A robotika témakör integrálásának lehetőségei a természettudományos tantárgyak oktatásában. InfoDidact 2019. Retrieved May 30, 2023, from <https://people.inf.elte.hu/szlavi/InfoDidact19/Manuscripts/GB.pdf>
- Grandell, L., Peltomäki, M., Back, R.J., & Salakoski, T. (2006). Why complicate things? Introducing programming in High School using Python. In: Tolhurst, D. and Mann, S., Eds. *Proc. Eighth Australasian Computing Education Conference (ACE2006)*. Hobart, Australia. CRPIT, 52. ACS, pp. 71–80.
- Donaldson T. (2003). *Python as a First Programming Language for Everyone*.
- Zelle, J.M. (1999). *Python as a First Language*.



B. Gaál – PhD student and assistant lecturer at ELTE Institute of Computer Science- Department of Media & Educational Technology. He is a member of the Hungarian Bebras Organization Team. Graduated with an MSc degree in teaching Geography and teaching Informatics. His main research areas are the integrability of robotics in the natural sciences and the usage of micro:bit in education.

Elementary Algorithms – Prefix Sum

László Gábor MENYHÁRT, László ZSAKÓ

Eötvös Loránd University, Budapest

e-mail: menyhart@inf.elte.hu, zsako@caesar.elte.hu

Abstract. In this paper we present a special problem, named as prefix sum, and its variations. We analyse the base problem more detailed and compare the possible solutions. Our students solved similar problems in competitions, so we have statistics about their solutions. We offer this article for those readers who are interested in programming methodology, or who are teachers, and their students will be participated in competitions.

Keywords: systematic programming, teaching, measurement, prefix sum, cumulative sum.

1. Introduction

During teaching of programming, after the introduction of algorithmic structures (sequence, condition, loop), basic algorithms (programming patterns, type algorithms), such as e.g. summary, search, maximum selection, sort, etc. are being dealt.

Problem-solving strategies appear in programming competitions (and in talent management) (e.g. greedy strategy, dynamic programming). However, the gap between the two is rarely filled. This was handled by J. Bentley's classic series of articles and book (Programming Pearls (Bentley, 1984)) from 1984, and more recently, for example, websites teaching algorithmization (Sannemo, 2018; ACM, 2023). Its first appearance was in 1954, in a magazine not directly on IT (Page, 1954).

In this article, we take a look at what the participants of Hungarian programming competitions do with the prefix sum (cumulative sum) task type. Interestingly, we are examining two very different age groups, in first students aged 13–16 can participate – Nemes Tihamér Online Programming Competition (Nemes, n.d.) (see later: Online), and in the other, university students who are studying computer science and just getting to know programming – Talent Search University Programming Competition (ELTE IK, n.d.) (see later: TUPC).

In this article, we do not intend to deal with advanced applications of the algorithm (balanced binary trees, Fenwick trees, ...).

2. Basic Task of Prefix Sum

Task: There is a series of numbers with N element in X and it contains both positive and negative numbers. Give the interval $[a, b]$ of series where the sum of the elements is maximal! We must define such an a and b indexes for which the following condition is satisfied:

$$1 \leq a \leq b \leq N$$

and

$$\forall p, q (1 \leq p \leq q \leq N): \sum_{i=p}^b X_i \geq \sum_{i=p}^q X_i$$

Solution: The basic solution calculates the sum of the numbers for each interval, and then selects the maximum for them:

```

sum(X, i, j) :
    S:=0
    For k=i to j
        S:=S+X[k]
    End for
    sum:=S
End function.

BasicSolution(N, X, a, b, max) :
    max:=-∞
    For i=1 to N
        For j=i to N
            s:=sum(X, i, j)
            If s>max then max:=s; a:=i; b:=j
        End for
    End for
End function.

```

Due to the three loops, its running time is proportional to N^3 . A frequently used idea is to first calculate not what the task asks for, but something simpler, from which it is easy to get the solution to the task.

Let's calculate the sum of the first i element of the series in the vector s .

$$\forall i (1 \leq i \leq N): s(i) = \sum_{j=1}^i X_j$$

which, using a recursive relation, is simpler:

$$\forall i (1 \leq i \leq N): s(i) = s(i-1) + X(i), \quad s(0) = 0$$

Then, the sum of the elements of an interval can be expressed as follows (Fig. 1):

$$\sum_{i=a}^b X_i = s(b) - s(a - 1)$$

Thus, the following algorithm – the method of cumulative summation – takes a step proportional to N^2 .

```
Cumulative sum(N, X, a, b, max) :
  s[0]:=0
  For i=1 to N
    s[i]:=s[i-1]+X[i]
  End for
  max:=-∞
  For i=1 to N
    For j=i to N
      If s[j]-s[i-1]>max then max:=s[j]-s[i-1]; a:=i; b:=j
    End for
  End for
End function.
```

3. Variations of Prefix Sum

This strategy can be used not only for summation (serial calculation), but also for other types of tasks. Basically, in cases where operations are associative (Blelloch, 1990), there is an “inverse” (Sannemo, 2018), so

$$f(a, b) = f^{-1}(f(1, b), f(1, a - 1)).$$

Index	0	1	2	3	4	5	6
Series		3	5	1	8	2	4
prefix sum	0	3	8	9	17	19	23
sum(1, 6)	0	3	8	9	17	19	23
sum(1, 2)	0	3	8	9	17	19	23
sum(3, 5)	0	3	8	9	17	19	23
							19-8=11

Fig. 1. prefix sum.

For example

$$\sum_{i=a}^b X_i = \sum_{i=1}^b X_i - \sum_{i=1}^{a-1} X_i$$

or

$$X_a * X_{a+1} * \dots * X_b = \frac{X_1 * X_2 * \dots * X_b}{X_1 * X_2 * \dots * X_{a-1}}$$

assuming that none of the values is 0.

However, the maximum selection task for the interval $[a, b]$ is not always interesting for the above, for example, for the sum it is uninteresting if all numbers are positive, and for the product, if all numbers are greater than 1.

Often the task is not maximum selection, but answering many questions for different intervals (Yao and Miao, 1990; USACO, 2015). In another type, we are looking for a certain type of sum, in the competition problem (USACO, 2016), for example, the longest series, where the sum of the numbers is divisible by 7.

Counting can also make sense for the item, but then either the length of the interval or some other property must be modified.

They can be:

- Let's give a sequence of at most K length, in which the number of elements with the given property is maximal!
- Let's give at most K long part of a series, in which the number of elements with the given property is maximal and the two extreme elements also have the given property!
- Let's give the longest part of a series in which at least half of the elements have a given property and the two extreme elements!

In many tasks, the method is included as a supplement, moreover, only if certain prerequisites are met.

Task: A random number generator generates numbers between 0 and $M-1$. We received the first N random numbers produced by the generator. Write a program to check the generator, it calculates for K pieces of interval $[A, B]$, how many numbers between 0 and $M-1$ do not occur in the given interval! The value of K is much greater than the value of M !

Solution: We can write a not so naive solution for it (it solves the counting by indexing):

```
None(N, X, K) :
  For i=1 to K
    In: A, B
    number[] := (0, ..., 0)
    For j=A to B
      number[X[j]] := number[X[j]] + 1
    End for
```

```

C:=0
For j=0 to M-1
  If number[j]=0 then C:=C+1
End for
Out: C
End for
End function.

```

According to this, its running time is proportional to $K * (N+M)$.

Cumulative summation first calculates in a matrix how many j values are in the first random number i , and then calculates the result from this:

```

None(N,X,K):
  number[0]:= (0,...,0)
  For i=1 to N
    number[i]:=number[i-1]; number[i, X[i]]:=number[i, X[i]]+1
  End for
  For i=1 to K
    In: A, B
    C:=0
    For j=0 to M-1
      If number[b, j]-number[a-1, j]=0 then C:=C+1
    End for
    Out: C
  End for
End function.

```

The first loop is $N*M$, and the second is $K*M$, even in the case of slow implementation of array evaluation. According to the task description, K is much larger than M , so $K*N+K*M$ is greater than $N*M+K*M$, and this is true if $K>M$, as stated in the task description.

4. Add an Extra Step

We can modify the task with extra steps. For instance, the solution of next task is more efficient with order.

Task: At an event, the number of male and female visitors is recorded for each day. Create a program that gives the period during which the total number of men and women was the closest to each other!

Solution: We can create the naïve solution when all time periods are calculated. Let $x[i]$ be i . the difference in the number of men and women per day! Running time $O(N^3)$.

```

min:=+∞
For i=1 to N
  For j=i to N
    S:=0
    For k=i to j

```

```

    S:=S+X[k]
  End For
  If |S|<min then mink:=i; minv:=j; min:=|S|
End For
End For

```

In the second version we calculate prefix amounts, and then take the smallest one from their difference. Running time $O(N^2)$.

```

s[0]:=0
For i=1 to N
  s[i]:=s[i-1]+X[i]
End For
min:= +∞
For i=1 to N
  For j=i to N
    If |s[j]-s[i-1]|<min
      then mink:=i; minv:=j; min:=|s[j]-s[i-1]|
    End For
  End For
End For

```

The optimal solution uses the difference between arranged and adjacent ones. The difference between two values in s is the smallest if they are closest to each other in value. Let's arrange them in a row, then take the one with the smallest difference from the adjacent ones! Running time $O(N \cdot \log_2 N)$.

```

s[0].db:=0; s[0].ind:=0
For i=1 to N
  s[i].db:=s[i-1].db+X[i]; s[i].ind:=i
End For
Order(s,0..N)
mini:=1
For i=1 to N-1
  If s[i+1].db-s[i].db<s[mini+1].db-s[mini].db then mini:=i
End For
min:=s[mini+1,1]-s[mini,1]
mink:=less(s[mini].ind,s[mini+1].ind)+1
minv:=greater(s[mini].ind,s[mini+1].ind)

```

Based on the test cases, we managed to achieve different scores for the different solution methods, so we can determine the distribution of solution types based on the scores achieved by the competitors (Fig. 2).

optimal	71–100
prefix sum	31–70
naïve	0–30

Fig. 2. Scoring of different solutions.

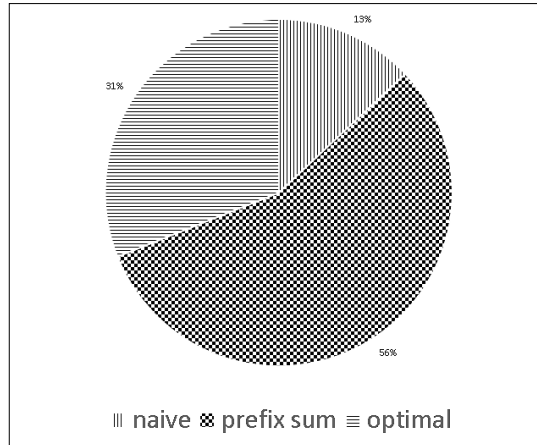


Fig. 3. Distribution of solution types.

Different scores within each group may be due to good or bad handling of special cases.

Fig. 3 shows the distribution of scores between each type of solution.

5. Mathematics and Matrices with Prefix Sum

A prefix sum problem can be generalized not only to vectors, but also to other structures, for example to a matrix as you can see in the following:

Task: A farmer wants to buy a rectangular plot of land in an $N \times M$ rectangle. He knows about each piece of land that can be bought, how much he would profit or lose if he cultivated it. Enter the rectangle on which the greatest profit can be achieved!

Solution: In the elementary solution, the sum of the elements of each (P, Q) upper-left and (R, S) lower-right sub-rectangle would have to be calculated, which would result in 6 loops, and then we could take their maximum (Fig. 4).

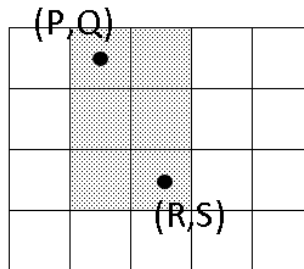


Fig. 4. The task.

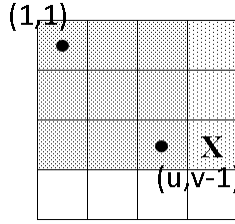


Fig. 5. Calculation of E.

Let's try to set a partial goal: calculate the value of the $(1, 1)$ upper left and (u, v) lower right corner tiles and store it in $E(u, v)$!

X = value of densely dotted rectangle + value of rarely dotted rectangle (Fig. 5)

```

E[1..N,0] := (0, ... 0)
For v=1 to M
  x:=0
  For u=1 to N
    x:=x+T[u,v]
  End for
  E[u,v] := E[u,v-1]+x
End for

```

Based on these, the sum of the values of the (i, j) upper left and (u, v) lower right rectangles is: $\text{value}(E, i, j, u, v) = E[u, v] - E[u, j-1] - E[i-1, v] + E[i-1, j-1]$. See the same on Fig. 6!

	...	j-1	j	...	v	...
...						
i-1	...	$+E_{i-1,j-1}$	$E_{i-1,j}$...	$-E_{i-1,v}$...
i	...	$E_{i,j-1}$	$E_{i,j}$...	$E_{i,v}$...
...						
u	...	$-E_{u,j-1}$	$E_{u,j}$...	$E_{u,v}$...
...						

Fig. 6. Calculation of result.

We write here the algorithm only so that we can modify it in the next step:

```

max:=-∞
For i=1 to N
  For j=1 to M
    For k=i to N
      For l=j to M
        o:=value(E,i,j,k,l)
        If o>max then P,Q,R,S:=i,j,k,l; max:=o
      End for
    End for
  End for
End for

```

The 4 loops are clearly visible, i.e. if N and M are of the same order of magnitude, then the running time is proportional to $O(N^4)$, to the fourth power of N .

In the competitions, we modified the task in the same way for both age groups, we fixed the area of the rectangle to be selected, i.e. we searched for four values for which there was a new condition that $(R-P+1) * (S-Q+1) = A$, for a given A constant, and the number of elements of some type had to be specified instead of the sum.

One of the tasks of the 3rd round of the competition held in 2017/18 was as follows: “In a rectangular area, we know the altitude above sea level of $N * M$ points. Points that are larger than their four neighbours are called vertices. There are certainly no peaks at the edges of the area. Write a program that gives you a rectangular area containing exactly A points, with as many vertices as possible!”

A significant part of the competitors, as expected based on the previous ones, gave the naive solution, with 6 loops. A significant part of those competitors who knew the method examined the size of the area within the above 4 loops:

```

If (k-i+1) * (l-j+1) = A then
  o:=value(E,i,j,k,l)
  If o>max then P,Q,R,S:=i,j,k,l; max:=o

```

Those who were a little more skilled could have saved one more loop and calculated the value of the variable l only for k values where A divided by $(k-i+1)$, but we hardly found such a solution.

```

max:=-∞
For i=1 to N
  For j=1 to M
    For k=i to N
      If A mod (k-i+1)=0 then
        o:=value(E,i,j,k,j+A div k-1)
        If o>max then P,Q,R,S:=i,j,k, j+A div k-1; max:=o
      End if
    End for
  End for
End for

```

In the most effective solution, the divisors of A can be calculated in `square_root(A)`¹ steps into a vector D with C elements, since the area can be calculated as the product of two sides.

Thus, the expected solution is $O(N^2C)$, taking advantage of the fact that $D[k] * D[C-k+1] = A$:

```
max:=-∞
For i=1 to N
  For j=1 to M
    For k=1 to C
      o:=value(E,i,j,i+D[k]-1,j+D[C-k+1]-1)
      If o>max then P,Q,R,S:=i,j,i+D[k]-1,j+D[C-k+1]-1; max:=o
    End for
  End for
End for
```

Based on the test cases, we managed to achieve different scores for the different solution methods, so we can determine the distribution of solution types based on the scores achieved by the competitors (Fig. 7).

Different scores within each group may be due to good or bad handling of special cases.

The university students use their more mathematical knowledge to score more points, and that they are moving from the naive and cumulative solution to a more efficient solution. They didn't even have a naive and simple cumulative solution. On the other hand, these together hardly reached the 50% rate in the 13–16 age group. However, the best competitors also found the optimal solution (Fig. 8).

6. Conclusion

The basic idea of the cumulative summation is therefore: Instead of a value calculated on an arbitrary subseries, we should only calculate a value for that subseries whose begin-

optimal, only with divider k , calculation of 1 from k	92–100
only with divider k , only with divider 1	71–91
calculation of k from 1	55–70
simple cumulative	41–54
naive	16–40
good with luck	0–15

Fig. 7. scoring of different solutions.

¹ It's enough to determine all divider till square root of A , because if x is divider of A , A/x is also divider.

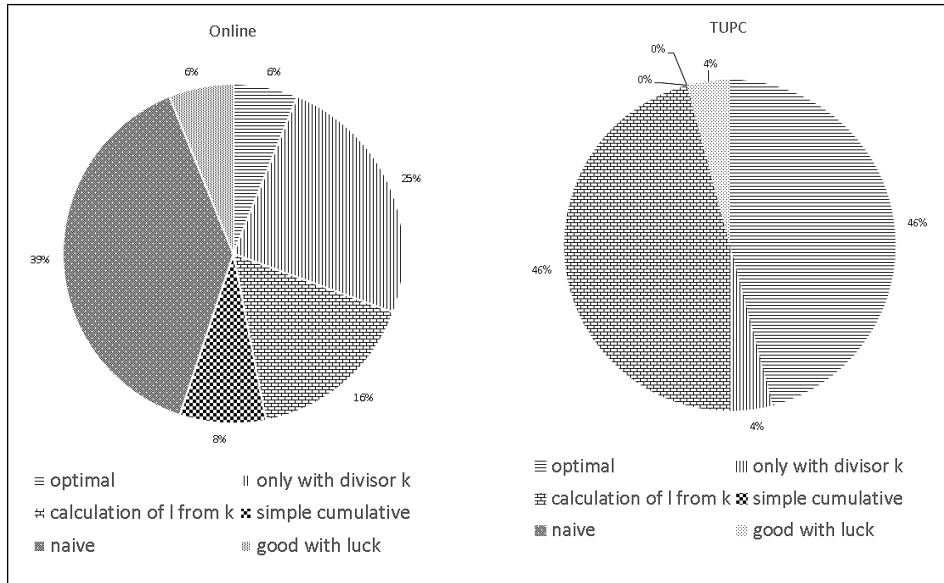


Fig. 8. Distribution of solution types.

ning is the beginning of the whole series, and then express the value calculated on the subseries from these!

The same principle can also be applied to calculation tasks instead of summation, i.e. the operation can be any associative operation whose inverse can be calculated.

Based on our experience in the competition, students learn the basic method easily, but they do not figure it out by themselves – typically, the younger we look at, the more the proportion of naive, slow solution makers increases. Only some of them notice the small modifications, e.g. when such sums need to be calculated both from the front and from the back, or another operation must be used instead of summation.

It is important to note that naive solutions include summation and maximum selection for most tasks, but these elementary algorithms (programming theorems) are also needed to write effective, cumulative summation. In other words, we are not talking about new algorithms, but we must think differently when applying the usual algorithms!

Mathematical considerations, as we saw in the last task, cause problems for students who are not strong in mathematics. And the math was just that if an integer is the product of two integers, then both numbers must divide the product – so it is necessary to produce the divisors. It can also be seen from the results that a significantly higher proportion of first semester university students solved the task with the maximum score.

Because of the above, we believe that in teaching programming, we should deal with the above basic method (avoiding all more complicated data structures). In this, teachers currently receive little help (Szlávi and Zsakó, n.d.), because the vast majority of scientific articles are about advanced solutions using different trees.

References

- ACM (2023). Category: Teaching kids programming. *Algorithms, Blockchain and Cloud*. (Accessed on: 2023.05.26.) <https://helloacm.com/category/teaching-kids-programming/>
- Bentley, J. (1984). Programming Pearls – Algorithm Design Techniques. *Comm. ACM*, 27(9), 865–871.
- Blelloch, G.E. (1990). *Prefix Sums and Their Applications*. Tech. Rep. CMU-CS-90-190. School of Computer Science, Carnegie Mellon University, Pittsburgh. (Accessed on: 2023.05.26.) <https://www.cs.cmu.edu/~guyb/papers/Ble93.pdf>
- ELTE IK (n.d.). Talent Search University Programming Competition. *Informatikai versenyek*. (Accessed on: 2023.05.26.) <http://versenyek.inf.elte.hu/versenyek/tehetsegkutato-egyetemi-programozasi-verseny>
- Nemes, T. (n.d.). *Online Programming Competition*. (Accessed on: 2023.05.26.) <http://tehetseg.inf.elte.hu/nemes-online/index.html>
- Page, E.S. (1954). Continuous Inspection Schemes. *Biometrika*, 41(1/2), 100–115. (Accessed on: 2023.05.26.) https://www.jstor.org/stable/2333009?origin=JSTOR-pdf&seq=1#metadata_info_tab_contents
- Sannemo, J. (2018). Principles of Algorithmic Problem Solving. (Accessed on: 2023.05.26.) <https://usaco.guide/PAPS.pdf#page=207>
- Szlávi, P., Zsakó, L. (n.d.) Elemi algoritmusok (a programozási tételek után). ELTE Informatikai Kar, Budapest. (Accessed on: 2023.05.26.) http://tehetseg.inf.elte.hu/szakkorefop2017/pdf/elteikszakkor_elemi_algoritmusok.pdf
- USACO (2015). *USACO 2015 December Contest, Silver, Problem 3. Breed Counting*. USA Computing Olympiad. (Accessed on: 2023.05.26.) <http://www.usaco.org/index.php?page=viewproblem2&cpid=572>
- USACO (2016). *USACO 2016 January Contest, Silver Problem 2. Subsequences Summing to Sevens*. USA Computing Olympiad. (Accessed on: 2023.05.26.) <http://www.usaco.org/index.php?page=viewproblem2&cpid=595>
- Yao, D., Miao, D. (n.d.). Introduction to Prefix Sums. (Accessed on: 2023.05.26.) <https://usaco.guide/silver/prefix-sums?lang=cpp>



L.G. Menyhárt is assistant professor at Department of Media & Educational Informatics, Faculty of Informatics, Eötvös Loránd University in Hungary. Since 2003 he has been involved in task creation of programming competitions in Hungary. His research interest includes teaching algorithms and data structures; didactics of informatics; methodology of programming in education; teaching programming languages; talent management.



L. Zsakó is a professor at Department of Media & Educational Informatics, Faculty of Informatics, Eötvös Loránd University in Hungary. Since 1990 he has been involved in organizing of programming competitions in Hungary, including the CEOI. He has been a deputy leader for the Hungarian team at IOI since 1989. His research interest includes teaching algorithms and data structures; didactics of informatics; methodology of programming in education; teaching programming languages; talent management. He has authored more than 68 vocational and textbooks, some 200 technical papers and conference presentations.

Indonesian Bebras Challenge 2021 Exploratory Data Analysis

Vania NATALI, NATALIA, Cecilia Esti NUGRAHENI

*Informatics Departments, Parahyangan Catholic University, Bandung, Indonesia
e-mail: vania.natali@unpar.ac.id, natalia@unpar.ac.id, cheni@unpar.ac.id*

Abstract. Indonesia is a full member of International Bebras Community and regularly perform the annual Bebras Challenge since 2016. The tasks for the Indonesian Bebras Challenge are taken and adapted from International Bebras Task pool, and each of the tasks is related to informatics and Computational Thinking (CT). We explore and analyze the Indonesian Bebras Challenge 2021 data to find interesting insight into Indonesian student's competencies in informatics and computational thinking, the suitability of the selected Bebras Tasks difficulty level for each age group, and the relation between participants' score and time duration in completing Bebras Challenge. The data exploratory has been done with statistics and data visualizations. We found that Indonesian students need to learn more about informatics and CT. The difficulty level of the Indonesian Bebras Challenge for elementary school students is still in accordance with Indonesian students' competency. In contrast, the Bebras Challenge difficulty level for junior and senior high school is higher than the students' competency. By analyzing the time duration of each participant completing the challenge, we also found some dishonest attitude presumptions during the online Bebras Challenge. We discover some suggestions for the improvement of Indonesian Bebras Challenge event as a means to improve Indonesian students' informatics and CT skills.

Keywords: Bebras Challenge, Bebras task, Exploratory data analysis.

1. Introduction

The Indonesian Ministry of Education, Culture, Research, and Technology (MoECRT) launched “Kurikulum Merdeka” (Independent Curriculum) as one of the curricula that are available in elementary, junior, and senior high schools in Indonesia (Direktorat Jendral PAUD Dikdas dan Dikmen, 2022). Before the implementation of Kurikulum Merdeka, informatics was an elective subject, and Computational Thinking (CT) is unrecognized in most of Indonesian education. In Kurikulum Merdeka, informatics has become a compulsory subject in junior and senior high school. For the elementary school, CT has integrated into Mathematics, Indonesian, Natural and Social Sciences subjects. CT has become a fundamental part of the informatics subject for junior and senior high school (MoECRT, 2021).

One of the challenges in Indonesian education is the inequality of education quality and IT facilities between big cities and remote areas (MoECRT, 2022). Indonesian students didn't get good results on the PISA Test (OECD, 2019). However, Indonesia still has excellent students who regularly participate in international Olympiads of various subjects, including Olympiads in Informatics (IOI), and got some achievements (IOI, 2023). Liem (2016) was said that although there is a gap in the quality of education between remote areas and big cities in Indonesia, it is hoped that the selection process for IOI will remain balanced between "potential candidates" from remote areas and "ready to compete" candidates from big cities.

Bebras Community is an international initiative that aims to promote informatics and computational thinking among students and teachers of all ages. One of the activities of Bebras community is organizing an annual contest called Bebras Challenge. Bebras task, the name of each question in Bebras Challenge, involves informatics and/or computational thinking concepts and must be possible to be answered without prior knowledge of informatics (Dagienė and Stupurienė, 2015).

Indonesia has been participating in organizing Bebras Challenge since 2016. The number of Indonesian Bebras Challenge participants in 2016–2021 is shown in Fig. 1.

The growing number of Indonesian Bebras Challenge participants in 2016–2021 (Bebras.org, 2011b) shows that Bebras is getting increasingly known in Indonesian education. Bebras Challenge become a good potential means to improve the informatics and CT skills of Indonesian students. Since Bebras Task can be answered without using a computer, it is suitable to be used to improve informatics skills for Indonesian students in remote areas with limited computer facilities.

Exploratory Data Analysis (EDA) is the act of looking at the data from many angles lookout for some interesting features. In short, EDA is looking at the data to see what it seems to say (Morgenthaler, 2009). Our data features are task code, task title, difficulty level, age group, CT and informatics concept, a unique identifier for each participant, time spent for each participant to complete the challenge, the score for each task, and the total scores of each participant.

Based on the fact about the challenges in Indonesian education, the potential that Indonesian students have, the new position of informatics subject and CT in Kurikulum



Fig. 1. Indonesian Bebras Challenge Participants Number.

Merdeka, and the growing number of Indonesian Bebras Challenge participants, the objectives of this Indonesian Bebras Challenge 2021 Exploratory Data Analysis (2021-ID-EDA) is exploring and finding interesting insights from Indonesian Bebras Challenge 2021 data. The EDA insights we discover are related to:

1. **(RQ1)** How appropriate is the difficulty level for each age group for students in Indonesia?
2. **(RQ2)** What is the general description of the informatics abilities of Indonesian students based on the results of the 2021 Bebras Challenge? Is there any specific kind of informatics or CT skill that need more attention for students in Indonesia?
3. **(RQ3)** Is there any interesting information about the participants' time competing in the Indonesian Bebras Challenge 2021? Is there any interesting relationship between the time spent on Bebras Challenge and the participants' grades? Are there any unreasonable data in the relation between spent time and participants' grades in the contest?

We hope that the result represents Indonesian students in the informatics and CT field.

The paper is organized as follows: Section 1 describes the introduction of this work, and in Section 2, we describe the literature review and we describe Indonesian Bebras Challenge 2021 characteristics in Section 3. In Section 4, we describe our EDA process. In Section 5, we report the 2021-ID-EDA results. Last, in the Section 6, we draw some conclusions and suggestions and describe our future works.

2. Literature Review

Bebras is an international initiative whose goal is to promote informatics and computational thinking, especially among teachers and students of all ages, but also to the public at large (Dagienė and Stupurienė, 2015). Currently, 55 countries join as full-member of Bebras International and 22 countries as provisional-member of Bebras (Bebras.org, 2023). Each country that joins Bebras is led by National Bebras Organization (NBO), that responsible for Bebras' activities within its country, such as creating and submitting Bebras Task, teacher training, organizing Bebras Challenge, selecting and translating Bebras Task so that the task become suitable for their country's Bebras Challenge (Dagienė and Stupurienė, 2016).

Bebras Challenge is a contest for elementary, junior, and senior high school pupils organized by International Bebras Community (IBC) and NBO in each country. The challenge is held on the second week of November, declared as worldwide Bebras Week. The main goal of Bebras Challenge is to motivate and encourage pupils to learn informatics fundamentals (concepts) and to support the development of algorithmic thinking and Computational Thinking. There are five age group recommendations from IBC, that are Little Beavers (grades 3–4), Benjamin (grades 5–6), Cadet (grades 7–8), Junior (grades 9–10), and Senior (grades 11–12). Each age group has its own question set.

Bebras task is a short question that involves fundamental informatics concepts and is answerable through the computerized interface. The criteria of Bebras task are solvable in 3 minutes, presentable on a single screen page, and independent from a specific system. Each task is equipped with recommendations about age group preferences and a difficulty level for each age group.

Some previous studies have analyzed Bebras Task and Bebras Challenge results. Bebras Challenge's result is significant to be analyzed because each country has a different standard of education. Bebras Task difficulty level needs to be assessed to ensure that the participants do not perceive it as appealing because it is too easy or too difficult (Bellettini, *et al.*, 2015). In (Bellettini, *et al.*, 2015), the difficulty level of the Bebras Challenge for Italian students has been analyzed using Item Response Theory (IRT). They found that 30% of the selected tasks were too easy or too difficult than the question preparation team expected. The results of this research can be used as input for the team compiling their questions in making adjustments for the next Bebras Challenge.

Izu, Mirolo, Settle, & Mannila (2017) analyzed the role of the CT concept in determining the difficulty level of questions, students' performance between schools in seven countries (Italy, Australia, Finland, Lithuania, South Africa, Switzerland, and Canada), and how gender differences affect scores. One part of their research was determining the CT concept that each Bebras Task addressed. Their research concludes that there are no significant differences in the CT skills of students from different schools. The CT concept does not determine the difficulty level of the questions. Differences in abilities for different genders are relative to age. At senior age, boys outperform girls in all countries.

3. Indonesian Bebras Challenge 2021

Even though Bebras Challenge is an international event, each country organizes its own Bebras Challenge. Bebras NBO can independently determine the age group of the pupils, select Bebras Tasks from the internationally accepted Bebras task pool, translate the task (Dagienė and Stupurienė, 2016), adjust the task's context based on the local situation, and determine the tools for their Bebras Challenge.

As we are in the pandemic situation, Indonesian Bebras Challenge 2021 was held fully online, and could be done at school or at home under the supervision of teachers. In multiple choice tasks, the participant gets plus points for the correct answer, 0 for the unanswered tasks, and negative points for wrong answers. While for short answer tasks, there is no penalty point for the wrong answer. Due to the limitation of Indonesian Contest Management System, Indonesia does not offer interactive task.

Indonesian Bebras Challenge is organized into four age groups. The name of the age category is inspired by the Indonesian scouting organization: Siaga (Cub Scout), Penggalang (Scouts), and Penegak (Rover Scouts). The youngest age group is named SiKecil (the small one). The age group, number of given tasks, and duration of the challenge are shown in Table 1.

Table 1
Indonesian Bebras Challenge Age Group

Age group name's / category	Student's age	Number of tasks	Duration of the challenge (minutes)
SiKecil	Up to 3 rd grade	8	30
Siaga	4 th –6 th grade	12	40
Penggalang	7 th –9 th grade	15	45
Penegak	10 th –12 th grade	15	25

Table 2
Number of Participants of Indonesian Bebras Challenge 2021

Age group name's / category	Number of Indonesian participants	Number of English participants	Total
siKecil	3.604	308	3.912
Siaga	5.800	348	6.148
Penggalang	9.558	582	10.140
Penegak	6.297	377	6.674
Total	25.259	1.615	26.847

The tasks selected for each age group are based on Bebras International age group and difficulty level preference. Each age group has a combination of task difficulty levels: easy, medium, and hard.

Since 2020, the Indonesian Bebras Challenge has been given in Indonesian or English. The English option has been offered as requested by some schools that adopt the international curriculum, primarily the school in big cities, where many of their students are interested in continuing their studies abroad. The number of participants for each category and language is shown in Table 2.

4. Exploratory Data Analysis Process

The 2021-ID-EDA process given in Fig. 2 is described below:

1. Data collection. The data are collected from two sources: (step 1) the Indonesian Bebras Challenge 2021 result and (step 2) the information related to each Bebras Task feature. Indonesian Bebras Challenge result was taken from the Bebras Challenge website with the permission of the Bebras Indonesia NBO. 33 files need to be combined before the data analysis process. The data features are a unique identifier for each participant, start time, end time, duration of the challenge, total score, and score for each task number.

The features of each Bebras Task are task code, task title, difficulty level, age group, and informatics concept. Those features are simply taken from the International Bebras Taks Workshop recommendation contained in the header

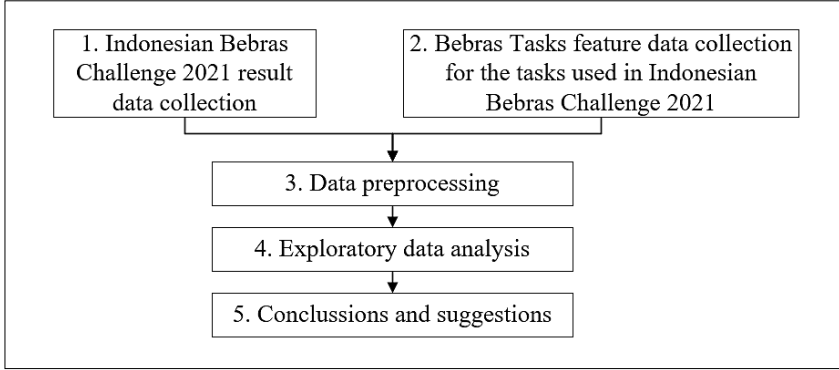


Fig. 2. Data Exploratory and Analysis Process.

of each task. We extracted other task features: question/task type and CT concepts involved in each task and the task language. We extracted manually CT concepts that were addressed in each task with a similar process that was done in (Izu, Mirolo, Settle, & Mannila, 2017).

2. Data preprocessing (step 3). The first step of data preprocessing is to combine the data from each age group category. There are three or four separate files for each age group (Table 1): the result of English participants, the result of Indonesian participants, and the make-up challenge for a specific language. We add a language column to the data.

For each task, there is a value that defines the score of that specific task for each participant. The maximum score is $100/(\text{number of tasks in a specific age group category})$. The minimum value is 0 for short answer questions and a negative point for the multiple-choice wrong answer. We do the data normalization by changing the score value with an integer 1 for the correct answer, -1 for the wrong answer, or 0 for the unanswered task. The normalization aims to categorize the participants' answers and calculate the percentage of each category.

3. The 4th and 5th steps are described in Sections 5 and 6.

The tool we used for this EDA is R Studio.

5. Exploratory Data Analysis

We did two points of view data exploration and analysis:

1. Score distribution for each task (Section 5.1). This exploration aims to answer RQ1 and RQ2.
 - a. To answer RQ1, we explore the mean, the standard deviation of participants' scores in each age group, and the success rate for each Bebras Task.

b. RQ2 is related to (Dagienė and Stupurienė, 2016) that the tasks should attract students and drive them to learn and explore to develop skills in a particular area. The exploration of each Bebras Task success rate can be analyzed for answering RQ2 as well.

3. Time spent completing Bebras Challenge (Section 5.2). This exploration aims to answer RQ3.

The threat of validity of this EDA is the data taken from all Indonesian Bebras Challenge 2021 participants. Due to the uneven education facilities and infrastructures (MoECRT, 2022) that may affect education quality in some areas of Indonesia, we could say that we use uncontrolled data.

5.1. Exploration and Analysis of Score Distribution for Each Age Group and Each Bebras Task Used in Indonesian Bebras Challenge 2021

The score distribution for each age group is shown in Fig. 3. Based on Fig. 3, we can see that the achieved score is lower in the older age group. Fig. 4 shows the score distribution for each age group and each Bebras Task language. There is no significant difference in score distribution between each task language. This finding is the first identification of no significant task language translation problem. However, the translation result will be analyzed further by each task analysis in Sections 5.1.1–5.1.4.

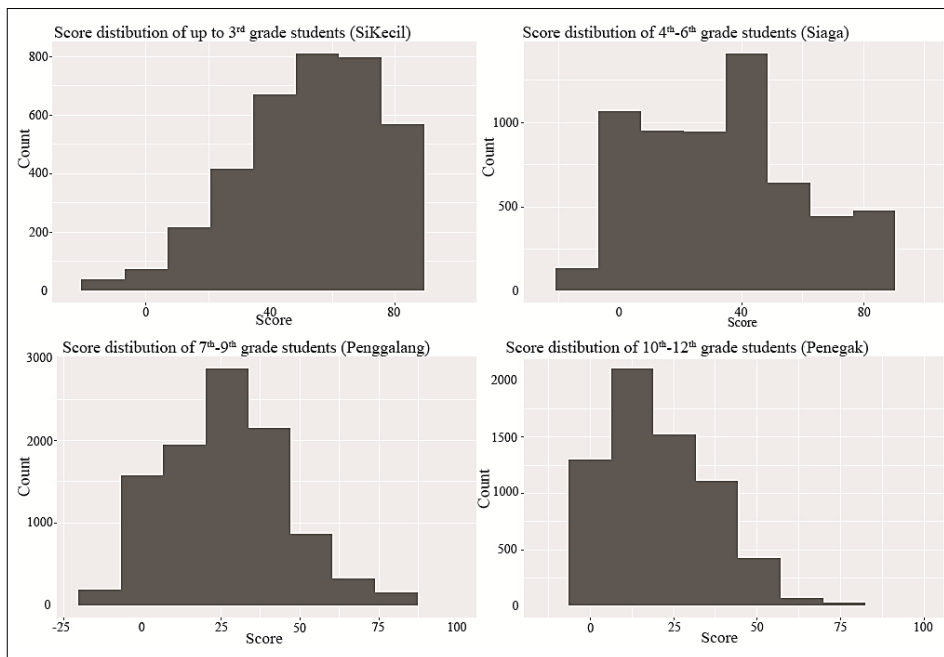


Fig. 3. Score Distribution for Each Age Group in Indonesian Bebras Challenge 2021.

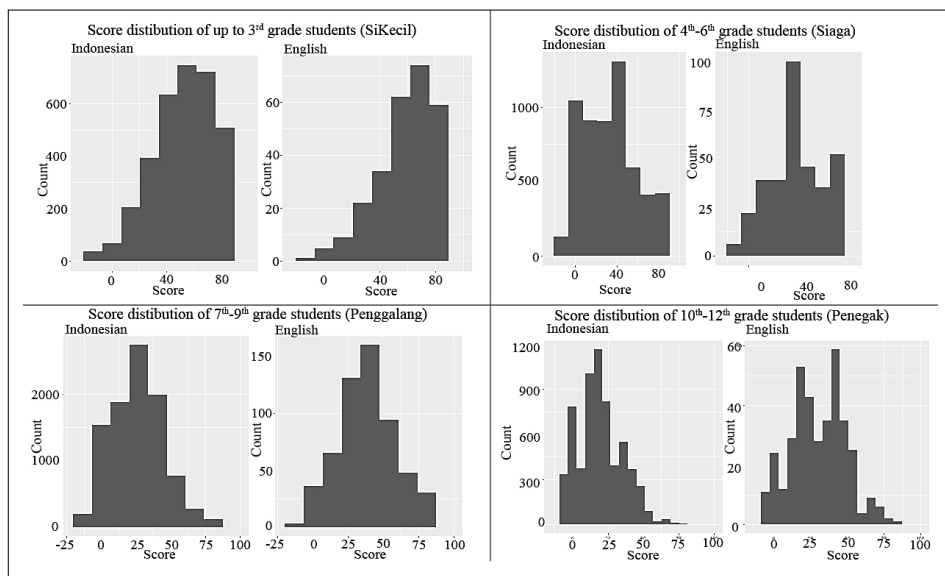


Fig. 4. Score Distribution in Indonesian Bebras Challenge 2021 for Each Age Group and Each Task Language.




The statistic of the Indonesian Bebras Challenge 2021 result is shown in Table 3. The maximum score for each participant is 100.

The statistics in Table 3 show that the mean score of English participants is higher than that of Indonesian participants and the mean scores are getting lower in the older age groups. There is no significant difference between the English and the Indonesian standard deviation. It means that score dispersions for both languages are similar. The

Table 3
Statistics of Indonesian Bebras Challenge 2021

Age Category and Statistic Components	Language	
	Indonesian	English
Up to 3 rd grade students (SiKecil)		
Score mean	55,66	63,64
Score standard deviation	26,78	25,65
4 th –6 th grade students (Siaga)		
Score mean	33,95	46,24
Score standard deviation	26,35	26,39
7 th –9 th grade students (Penggalang)		
Score mean	25,44	39,37
Score standard deviation	19,46	22,09
10 th –12 th grade students (Penegak)		
Score mean	19,39	30,48
Score standard deviation	15,90	18,63

Table 4
Bebras Task Answer Category for Fig. 5–Fig. 8

Category number	Category color	Meaning
1	 blue	Correct answer
0	 green	Blank
-1	 pink	Wrong answer

Indonesian translation for Bebras Task tends to be difficult because some common terms in English, especially informatics terms, do not have the Indonesian standard word translation. Thus, sometimes the sentences in Indonesian Bebras Task are longer than in English, so the students need more time to read the task.

In Sections 5.1.1–5.1.4, Indonesian students' performance is analyzed by their success rate in answering each Bebras Task. The success rate describes the students' ability to answer each task correctly. The answer category of each Bebras Task is displayed in different colors and category numbers, described in Table 4.

Each task answer category distribution is analyzed with the information about the task: difficulty level for each age group recommendation, task type, informatics concepts, and CT concepts. Here are the abbreviations used in each Indonesian Bebras Task 2021 description:

- Difficulty level: Hard (H), Medium (M), Easy (E).
- Task type: Multiple choices (MC) and Short answer (SA).
- Informatics concepts: Algorithms and programming (AP); Data, data structures, and representations (DSR); Interactions, systems, and society (ISC); Computer processes and hardware (CH); Communication and networking (CN).
- CT concepts: Abstraction (A); Algorithmic thinking (AT); Decomposition (D); Evaluation (E); Pattern recognition (P).
- Language: English (en); Indonesian (id).

5.1.1. Students up to 3rd Grade Age Group Performance (Category: SiKecil)

The Indonesian Bebras Challenge 2021 answer category distribution for the SiKecil Category for each language is shown in Fig. 5.

The color and category explanation for Fig. 5 is given in Table 4. Based on Fig. 5, the success rate of Indonesian and English participants in task number 8 differs quite a lot. The success rate of Indonesian and English are 44,9% and 24%, respectively. Thus, the difference is 20%. This information can be a reason for reviewing the Indonesian task translation.

To review the suitability of the international difficulty level suggestion of the Bebras Task and the competency of the SiKecil age group, we analyze each task's success rate. The Bebras task list for the SiKecil category is given in Table 5. The grey rows in Table 5

refer to tasks with a success rate of less than 50%. It means more than half of the students cannot answer the task correctly.

Based on Fig. 5 and Table 5, the students didn't perform well in Bebras Tasks number 2, 6, and 8, which have a hard difficulty level. The worst students' performance was on task number 8, which has a hard difficulty level and task type as short answer question. The characteristics of CT concepts on tasks 2, 6, and 8 combine algorithmic thinking and another concept. The tasks that have an easy and medium difficulty level can be answered correctly by most Indonesian participants. The difficulty level for the SiKecil category is generally suitable for Indonesian students.

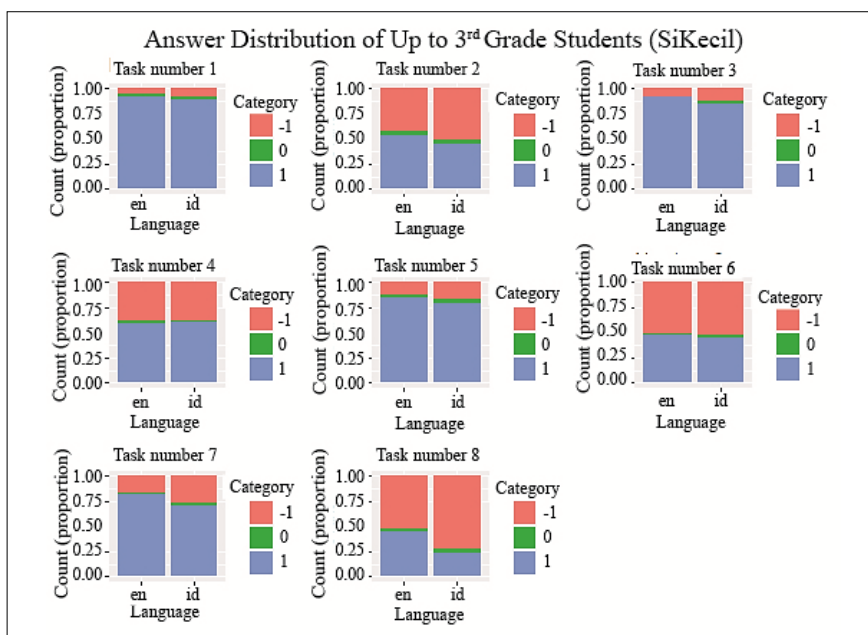


Fig. 5. Answer Distribution for up to 3rd Grade Students (SiKecil).

Table 5
Bebras Task List for up to 3rd Grade Students (SiKecil)

Number	Task Code	Task Title	Success rate	Difficulty Level	Task Type	Informatics Concepts	CT Concepts
1	2021-CN-01	The Lost Gold	89,34%	M	MC	DSR	AT
2	2021-CN-06a	Forgetful Little Beaver	46,37%	H	MC	AP	AT, E
3	2021-IE-05	Dancing Dress	85,69%	E	MC	AP, DSR	E
4	2021-LT-06	Do They Meet	59,94%	H	MC	AP	AT
5	2021-PK-09a	Creatures	80,44%	E	MC	AP, DSR	E, D
6	2021-SA-02	Flower Growth Phase	44,86%	H	MC	AP	AT, A
7	2021-UY-01	Circular Beaver	72,11%	M	MC	DSR	A
8	2021-UY-06	Fruit Road	25,72%	H	SA	DSR	AT, E

5.1.2. 4th–6th Grade Age Group Students' Performance (Category: Siaga)

The color and category explanation for Fig. 6 is given in Table 4. Based on Fig. 6, task number 3, 4, 5, and 9 have success rates that differ quite a lot between English and Indonesian participants. The success rates are:

- Task number 9: Indonesian: 34,6%; English: 77,2%; difference: 42,6%.
- Task number 5: Indonesian: 34,6%; English: 71,8%; difference: 37,2%.
- Task number 3: Indonesian: 45,3%; English: 64%; difference: 18,7%.
- Task number 4: Indonesian: 40,3%; English: 56%; difference: 15,7%.

The differences in the success rates can be a suggestion to review the task translation or context for Indonesian students.

The suitability between the international difficulty level suggestion of the Bebras Task and the competency of the Siaga category participants has been made in the same way as the SiKecil age group. The grey rows in Table 6 have the same meaning as in Table 5.

The retrieved information from Fig. 6 and Table 6 are:

- The lowest participants' success rate is at task number 1 (2021-CA-01, Cuckoo Birds), which has a hard difficulty level. The Indonesian and English success rates are 23,6% and 34,7%, respectively.
- The highest participants' success rate is task number 10 (2021-PK-09, Creatures), which has an easy difficulty level. The Indonesian and English success rates are 69,5% and 55,1%, respectively.

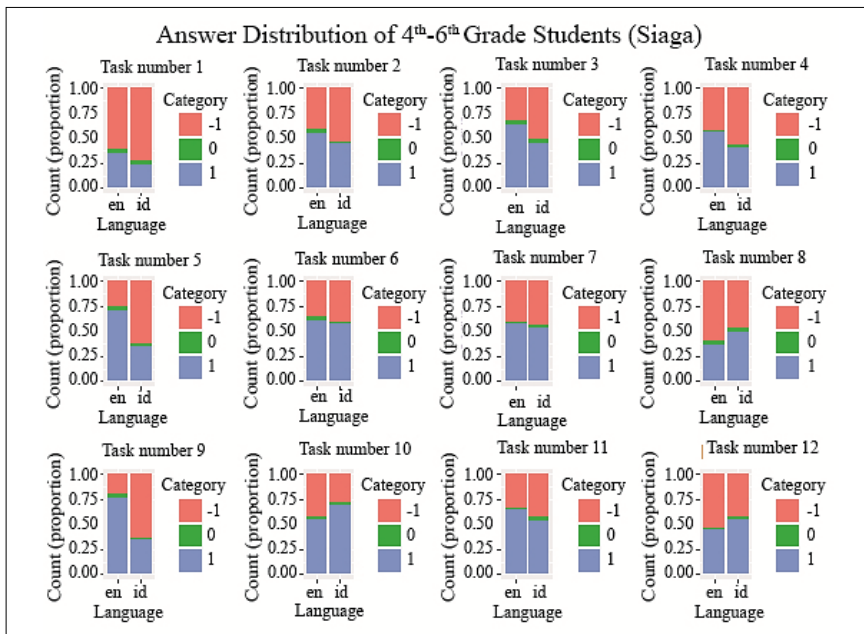


Fig. 6. Answer Distribution for 4th–6th Grade Students (Siaga).

Table 6
Bebras Task List for 4th–6th Grade Students (Siaga)

Number	Task Code	Task Title	Success Rate	Difficulty Level	Task Type	Informatics Concepts	CT Concepts
1	2021-CA-01b	Cuckoo Birds	24,32%	H	MC	DSR	E
2	2021-CN-03a	Picking Up Carrots	44,65%	M	MC	AP, DSR	AT, E
3	2021-CN-06	Forgetful Little Beaver	46,39%	H	MC	AP	AT, E
4	2021-IE-01	Presents Program	41,20%	H	MC	AP	A, E, AT
5	2021-IE-02	Coin Bag	36,73%	M	MC	DSR	A, D
6	2021-KR-01	Gujeolpan	57,03%	M	MC	DSR	P, E
7	2021-KR-02	Moving the Balls	52,80%	M	MC	AP, DSR	AT
8	2021-LT-06	Do they Meet	49,41%	M	MC	AP	AT
9	2021-LT-07	Find a Mistake	37,09%	M	MC	AP	AT, E
10	2021-PK-09	Creatures	68,69%	E	MC	AP, DSR	E, D
11	2021-RO-02	Volcanos	54,68%	H	MC	CN	AT, E
12	2021-UY-01	Circular Beaver	54,21%	E	MC	DSR	A

- c. 7 of 12 tasks got success rates less than 50%.
- d. More than 50% of participants can correctly answer both easy tasks.
- e. 2 of 6 medium tasks can be correctly answered by more than 50% of participants.
- f. 1 of 4 hard tasks can be correctly answered by more than 50% of participants.
- g. Five tasks combining Algorithmic Thinking and Evaluation CT concepts (tasks no. 2, 3, 4, 9, 11). The success rates for four of them are less than 50%.

Based on points a to f, the difficulty level of the twelve Bebras tasks in the Siaga category is still in accordance with the competence of Indonesian students.

5.1.3. 7th–9th Grade Age Group Students' Performance (Category: Penggalang)

The color and category explanation for Fig. 7 is given in Table 4. Based on Fig. 7, the success rates of Indonesian and English participants differ quite a lot on task number 5 and 6. The differences are:

- a. Task number 5: Indonesian: 46,7%; English: 68,3%; difference: 21,6%.
- b. Task number 6: Indonesian: 42,2%; English: 60,6%; difference: 18,4%.

The differences in the task's success rate may have two meanings, the quality of the students is different or can be a suggestion to improve the task translation or context for Indonesian students.

The grey rows in Table 7 have the same meaning as in Table 5. The retrieved information from Fig. 7 and Table 7 are:

- a. 12 of 15 tasks cannot be correctly answered by more than 50% of participants in the Penggalang category.
- b. 2 of 3 easy tasks can be answered correctly by more than 50% of participants in the Penggalang category. The easy task difficulty level is still in accordance with Penggalang category participants.

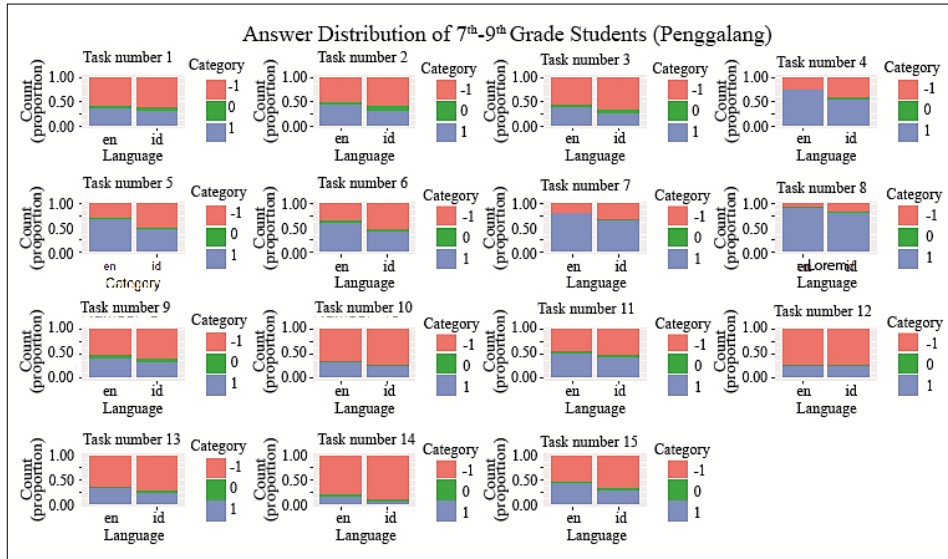


Fig. 7. Answer Distribution for 7th–9th Grade Students (Penggalang).

Table 7
Bebras Task List for 7th–9th Grade Students (Penggalang)

Number	Task Code	Task Title	Success Rate	Difficulty Level	Task Type	Informatics Concepts	CT Concepts
1	2021-BE-03	Necklace Instruction	31,70%	M	MC	AP, DSR	A, P
2	2021-CA-02	Spider Quilt	31,95%	H	MC	DSR	AT, D
3	2021-CA-04	Line of Fish	26,58%	M	MC	AP, DSR	AT, E
4	2021-CH-04c2	Strawberry Thief	55,64%	E	MC	AP, DSR	A, E
5	2021-CN-02	Maze	47,99%	E	MC	AP	AT
6	2021-DE-07	Turtle Path	43,26%	M	MC	AP	A, AT
7	2021-ID-10	Density of Liquid	64,42%	M	MC	DSR	E, AT
8	2021-IS-04a	Between Dots	80,33%	E	MC	AP, DSR	AT
9	2021-IT-1b	Strange Sorting	31,83%	H	MC	AP, DSR	E, AT
10	2021-LT-01	Meeting race	23,78%	H	MC	AP, ISC	AT
11	2021-LT-05	Compare	43,19%	H	MC	AP	AT, E
12	2021-NZ-01	Hidden Chocolate	23,28%	H	MC	AP	AT, E, D
13	2021-UZ-02	Pruning the Tree	24,47%	M	MC	AP, DSR	AT, D
14	2021- LT-03	Three Beavers	5,93%	M	SA	AP, CH	A, AT
15	2021-AT-01	Forest Observation	28,81%	M	SA	AP	A, AT

- c. The lowest participants' success rate is task number 14 (2021-LT-03, Three Beavers). The task type is a short answer question. It needs an integer as the input, so there is a low possibility of a wrong formatting answer or typo.
- d. The participants' success rates for short answer tasks (2021-LT-03 and 2021-AT-01) are meager.

- e. Due to the low success rate in general, the CT and Informatics concepts analysis cannot be done.

Based on points a to e, the performance of the Penggalang category participants is lower than the expectation of the International Bebras Committee, which is stated in each task difficulty level.

5.1.4. 10th–12th Grade Students' Performance (Category: Penegak)

The color and category explanation for Fig. 8 is given in Table 4. Based on Fig. 8, five tasks cannot be answered correctly by most of the participants. The participants' success rates are:

- Task number 12: Indonesian: 11,70%; English: 3,18%; in general: 1,29%.
- Task number 14: Indonesian: 3,03%; English: 7,16%; in general: 3,26%.
- Task number 11: Indonesian: 3,76%; English: 8,48%; in general: 4,03%.
- Task number 13: Indonesian: 10,60%; English: 22,01%; in general: 11,32%.
- Task number 10: Indonesian: 11,95%; English: 26,79%; in general: 12,79%.

Task 15 is the task that has the most difference in English and Indonesian success rates. The success rate for Indonesian participants is 42,57% and for English participants is 63,12%; thus, the difference is 20,55%.

The grey rows in Table 8 have the same meaning as in Table 5.

The retrieved information from Fig. 8 and Table 8 are:

- 13 of 15 tasks cannot be correctly answered by more than 50% of participants in the Penegak category.
- The participant's success rate for tasks number 1 and 8 are also under 60%.
- 4 of 5 tasks with the lowest success rate are short answer questions.

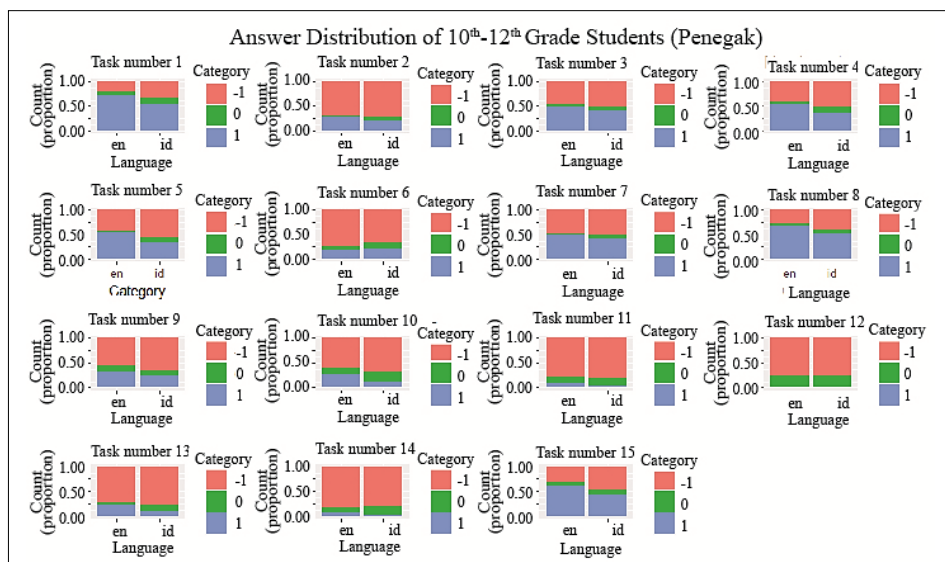


Fig. 8. Answer Distribution for 10th–12th Grade Students (Penegak).

Table 8
Bebras Task List for 10th–12th Grade Students (Penegak)

Number	Task Code	Task Title	Success Rate	Difficulty Level	Task Type	Informatics Concepts	CT Concepts
1	2021-AT-04d	Hashing	55,38%	M	MC	AP, DSR	AT
2	2021-BE-02	Vaccination Center	21,56%	M	MC	AP, DSR	E, AT
3	2021-CH-26b	Chez Connie	41,95%	E	MC	CH	AT
4	2021-NZ-02	Ice Cream Machine	36,20%	H	MC	CH	AT, E, D
5	2021-PT-05	Flooding	36,86%	E	MC	AP, DSR	A, AT
6	2021-RU-01	Save the Trees	21,82%	M	MC	AP, DSR	AT, E
7	2021-SK-02	Lift	43,66%	E	MC	DSR	D, A
8	2021-SP-01	Ordering 7 Students	53,27%	M	MC	AP	AT
9	2021-TW-03b	Napping Together	24,60%	E	MC	DSR	AT, A
10	2021-DE-04	Lawn Mover	12,80%	E	SA	AP, DSR, ISC	AT
11	2021-EE-03	Logs	4,03%	M	SA	AP, CN	AT, P, A
12	2021-IT-05	Snow White	1,29%	M	MC	AP	AT, E
13	2021-LT-01	Meeting Race	11,33%	M	SA	AP, ISC	AT
14	2021-PH-03	Great Wall of Beavaria	3,27%	M	SA	AP	AT
15	2021-CH-06	Compact Representation	43,74%	H	SA	DSR	AT

- d. All the easy tasks get a success rate under 50%.
- e. As in the Penegak category, the CT and Informatics concepts analysis cannot be done due to the low participants' success rate in general.

Based on points a to e, the performance of the Penegak category participants is lower than the expectation of the International Bebras Committee, which is stated in each task difficulty level.

5.2. Exploration and Analysis of Time Spent Completing Indonesian Bebras Challenge 2021

The purposes of the analysis of time spent completing the Indonesian Bebras Challenge 2021 are to know the majority of the time needed by Indonesian Bebras Challenge participants in completing the challenge and identify whether there is unreasonable data found during the challenge. We assume that the unreasonable data of time spent behavior is when the participant got a high score in less than the given duration for each age group category. The assumption was taken because we assume it is very difficult for the student to solve each task in one and a half minutes since they need time to read and understand the problem before thinking about its solution. We realize that this assumption may not be true for a very clever student.

Fig. 9 shows the time distribution for each Indonesian Bebras Challenge age group participant to complete the challenge. The graphic shows that the older the age group is, the longer time needed to finish the challenge. There are some outliers in Penggalang and Penegak categories.

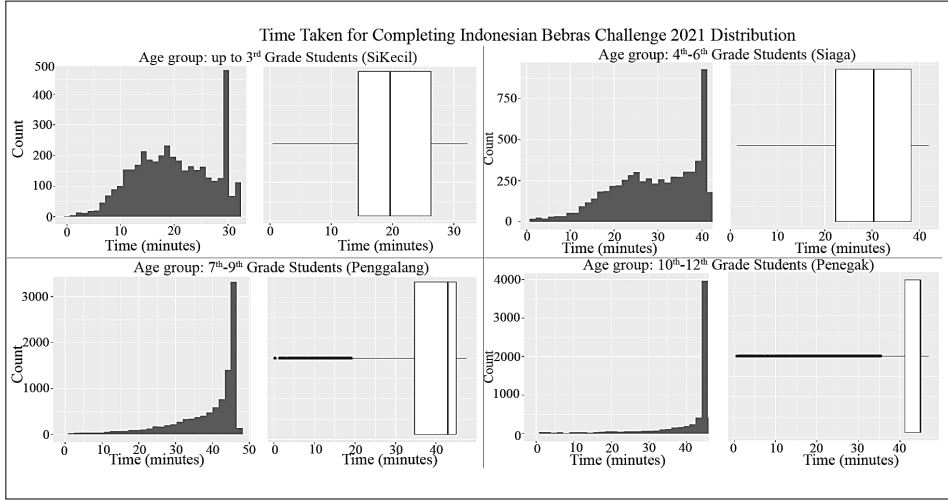


Fig. 9. Time Taken for Completing Indonesian Bebras Challenge 2021 Distribution for Each Age Group.

To gain more information, we explore the relationship between the time taken to complete Indonesian Bebras Challenge 2021 and the participant's scores for each age group. The plot results are shown in Fig. 10. There are two unreasonable data based on Fig. 10:

- i. The blue boxes in Fig. 10, show the scores from participants who completed the challenge in more than the given duration. It was confirmed to the Indonesian Bebras Challenge 2021 organizing committee that those were the consequences of the server's queue due to the auto-submit system when the challenge's time was up.
- ii. The red boxes in Fig. 10 show the mapping between participants who got high scores in less than half of the given challenge duration for each age group. These unreasonable data indicate the possibility that some participants behaved dishonestly during the challenge.

Fig. 10 shows that some participants finished the challenge in a short time. We looked closer and made the plots to observe whether the low participants' scores resulted from the perfunctory participants. We assume that the perfunctory participants were guessing the answer without thinking carefully to get the correct answer. The plot is shown in Fig. 11.

We took data from 25% of the participants who got the lowest score in each age group and drew the chart of their time taken in finishing the challenge in Fig. 11. Those graphics show that most participants with low scores still worked for over half the duration. Some of them finished the challenge within the maximum given duration. This information may lead to the conclusion that not all participants with low scores are perfunctory. They still gave their effort until the time was up.

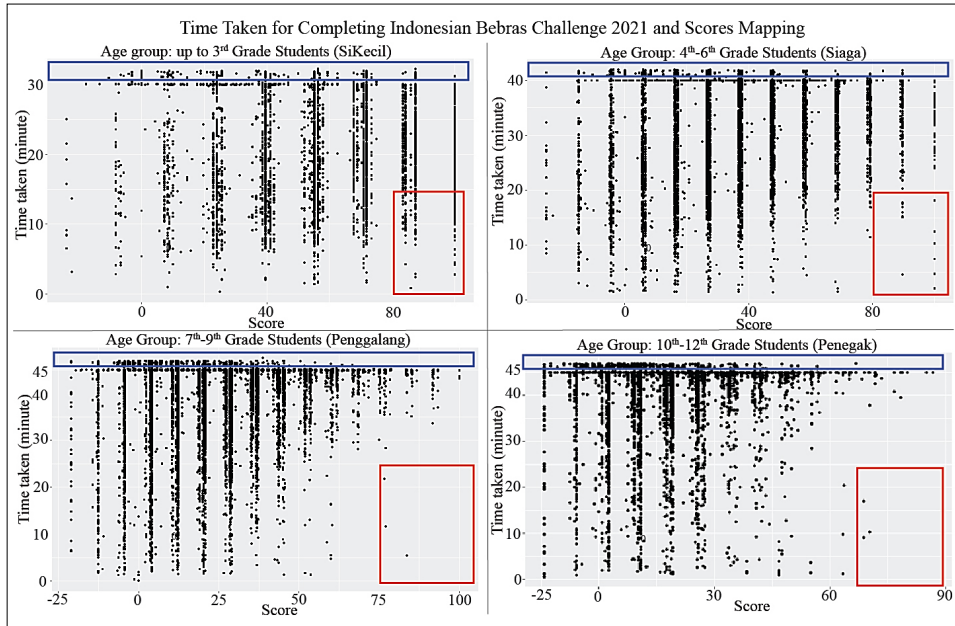


Fig. 10. Time Taken for Completing Indonesian Bebras Challenge 2021 and Scores Mapping.

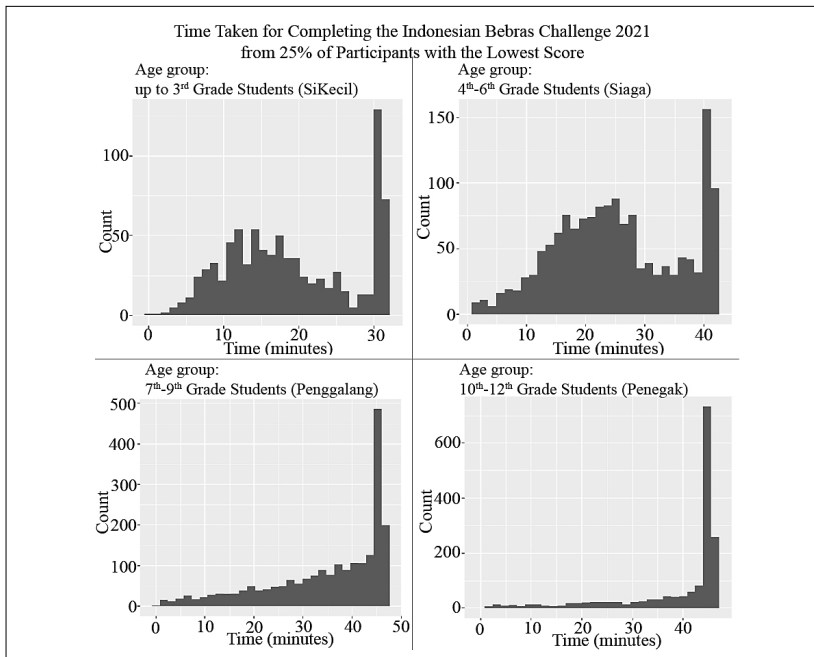


Fig. 11. Time Taken for Completing the Indonesian Bebras Challenge 2021 from 25% of Participants with the Lowest Score.

5.3. The Result of Indonesian Bebras Challenge 2021 EDA

The result of the Indonesian Bebras Challenge 2021 EDA is given by answering each EDA question (**RQ1–RQ3**).

1. **Respond to RQ1.** Based on Fig. 5–Fig. 8 and Table 5–Table 8, the difficulty levels of selected Bebras Tasks for SiKecil and Siaga categories are still in accordance with Indonesian students' competence. Due to the many questions that cannot be answered correctly by more than 50% of participants, it can be concluded that the difficulty level of Bebras Tasks in Penggalang and Penegak categories as stated by the international Bebras Task Committee are higher than the Indonesian participants' competence.

In general, the participants of SiKecil (up to 3rd-grade students) and Siaga (4th–6th-grade students) categories perform quite well in the Bebras Challenge. The participants of Penggalang (7th–9th grade students) and Penegak (10th–12th grade students) categories did not perform well enough. The score mean of each age group given in Table 3 shows that the younger age group is higher than the older age group.

2. **Respond to RQ2.** In general, Indonesian students' informatics competency still needs improvement. Based on Table 5, the informatics concepts in selected Bebras Tasks for the SiKecil category are not varied enough to be the base of informatics skill analysis. The participants of the SiKecil category need more exercise on the task that involves an Algorithmic Thinking concept combined with another concept. Based on Table 6, Table 7, and Table 8, the analysis of participants' informatics and CT skills cannot be done due to the low success rate in many tasks.
3. **Respond to RQ3.** Based on Fig. 9, it can be concluded that the higher the challenge category was, the longer time needed to complete the challenge. Many participants finished the challenge in the maximum given time in Penggalang and Penegak categories. We also found two unreasonable data based on the plot of the relation between the time needed to complete the challenge and the participants' score as shown and described in Fig. 10. Some dishonest attitude presumptions during the online Bebras Challenge event can be caught in Fig. 10: some participants got high scores in a short completion time.

The suggestion for the next Indonesian Bebras Challenge events based on this EDA are:

- a. The method in selecting Bebras Task for each age group in the Indonesian Bebras Challenge must be discussed in the Indonesian Bebras Challenge task preparation team, especially for Penggalang and Penegak categories which have low success rates in many tasks. Bebras Task difficulty level needs to be assessed to ensure that the participants do not perceive it as appealing because it is too easy or difficult (Bellettini, *et al.*, 2015). The task preparation teams need to adjust the difficulty level of each age group as suggested by the International Bebras Task Workshop

since Indonesia still has low PISA Test scores compared to other countries that join PISA Test, that shows, in general, the Indonesian students' performance is lower than many other countries. The fact that Informatics and CT are new in Indonesian education curriculum may affect the Indonesian students' performance in Indonesian Bebras Challenge.

- b. The Indonesian Bebras Challenge preparation teams may ask teacher review for the Indonesian tasks translation so that the translation is appropriate for each age group.
- c. The Indonesian Bebras Challenge preparation teams need to adjust the rules or the system related to the online Bebras Challenge event. The monitoring system of the online event needs to be evaluated to minimize the dishonesty that happened in Indonesian Bebras Challenge. Indonesian Bebras NBO needs to work with the teacher to encourage the students to work independently.

6. Conclusion

The 2021-ID-EDA questions (RQ1–RQ3) has been answered by the EDA result. By exploring the data related to Indonesian students' performance in Indonesian Bebras Challenge 2021 and the evaluation of Bebras tasks difficulty level for each age group, we found that the task difficulty level for the elementary students age group is still in accordance with the Indonesian student's competencies. But the junior and senior high school students did not perform well in the Indonesian Bebras Challenge 2021. The difficulty level of selected Bebras Tasks for Penggalang and Penegak categories is higher than the students' competencies. The unreasonable data was found by analyzing the relation between the time for completing the challenge and the participants' scores that led to a presumption of students' dishonest attitude in the online Bebras Challenge event. The older the age group, the longer time needed for the students to finish the challenge. In general, Indonesian students still need a lot of work to increase their skills in informatics and CT.

We also discover three suggestions for the Indonesian Bebras Challenge preparation team to improve the Indonesian Bebras Challenge event. The suggestions are about (1) the selection of Bebras Task difficulty level for each age group, (2) the improvement of Indonesian task translation, and (3) the evaluation of the monitoring system of the online Indonesian Bebras Challenge event. We hope that the results of 2021-ID-EDA can make the Bebras challenge implemented better in Indonesia so that Indonesian students can practice and increase their knowledge in informatics and CT.

Our future work will analyze the Indonesian Bebras Tasks translation used in SiKecil and Siaga (up to 6th-grade students) categories. There is some feedback from the teachers and parents that their child cannot understand some terms used in the tasks. This research will be done together with elementary school teachers.

Acknowledgments

The authors would like to thank Inggriani Liem, the head of Indonesian NBO, and Adi Mulyanto, the leader of the organizing committee of Indonesian Bebras Challenge for their support and the permission to use the data for this research.

References

- Bebras.org (2011a). *Countries*. Retrieved February 17, 2023, from Bebras: International Challenge on Informatics and Computational Thinking: <https://www.bebas.org/countries.html>
- Bebras.org (2011b). *Statistics*. Retrieved December 2022, 2021, from Bebras: International Challenge on Informatics and Computational Thinking: <https://www.bebas.org/statistics.html>
- Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M. (2015). How Challenging are Bebras Tasks? An IRT analysis based on the performance of Italian students. *The 2015 ACM Conference*. DOI: 10.1145/2729094.2742603
- Dagienė, V., Sentance, S. (2016). It's computational thinking! Bebras tasks in the curriculum. In: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*.
- Dagienė, V., Stupurienė, G. (2015). *Informatics Education based on Solving Attractive Tasks through a Contest*.
- Dagienė, V., Stupurienė, G. (2016). Bebras – a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 15(1), 25–44.
- Direktorat Jendral PAUD Dikdas dan Dikmen (2022, February 12). *Luncurkan Kurikulum Merdeka, Mendikbudristek: Ini Lebih Fleksibel!* Retrieved January 17, 2023, from Direktorat Sekolah Dasar: <https://ditpsd.kemdikbud.go.id/artikel/detail/luncurkan-kurikulum-merdeka-mendikbudristek-ini-lebih-fleksibel>
- Izu, C., Mirolo, C., Settle, A., Mannila, L. (2017). Exploring Bebras tasks content and performance: A multinational study. *Informatics in Education*, 39–59. DOI: 10.15388/infedu.2017.03
- Liem, I. (2016). Reshaping Indonesian students training for IOI. *Olympiads in Informatics*, 195–205.
- MoECRT (2021). *Kebijakan Kurikulum untuk Membantu Pemulihan Pembelajaran*. Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi.
- MoECRT (2022). *Kajian Akademik: Kurikulum untuk Pemulihan Pembelajaran*. Indonesia: Pusat Kurikulum dan Pembelajaran Badan Standar, Kurikulum, dan Asesmen Pendidikan Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi.
- Morgenthaler, S. (2009). Exploratory data analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 33–44.



V. Natali earned her bachelor's degree from the Department of Computer Science, Parahyangan Catholic University (UNPAR), and master's degree from Dept. of Informatics Engineering, Bandung Institute of Technology (ITB), Bandung, Indonesia. In 2017, she began to be involved in Computational Thinking and Informatics education with Bebras Indonesia and became a member of the Scientific Committee of Bebras Indonesia. She became Indonesia's representative at the International Bebras Task Workshop in 2022 and 2023, and in charged as Bebras Task Preparation team for Indonesian Bebras Challenge. She was involved in a writing team for Informatics textbooks for 7th–9th grade school and Computational Thinking module for Teacher Profession Program published by the Ministry of Education, Culture, Research and Technology of the Republic of Indonesia.



Natalia studied at the Mathematics Department of Parahyangan Catholic University (UNPAR) and earned her bachelor's degree in 2012. She took Mathematics Master program at the Bandung Institute of Technology. Currently, she is a lecturer in the Department of Informatics at Parahyangan Catholic University. Since 2017, Natalia has been actively involved in community service in Computational Thinking and Informatics education with Bebras Indonesia. She became Indonesia's representative at the 2023 International Bebras Task Workshop and in charged as Bebras Task Preparation team for Indonesian Bebras Challenge. She was part of the writing team for the 8th and 9th grades Informatics textbooks and Computational Thinking module for the Teacher Professional Program published by the Ministry of Education, Culture, Research and Technology of the Republic of Indonesia.



C.E. Nugraheni received her bachelor's degree (1993) and master's degree (1995) from Dept. of Informatic Engineering, Bandung Institute of Technology (ITB), Bandung, Indonesia. She received her doctoral degree (2004) from Dept. of Informatics, Ludwig Maximilians Universität, Munich, Germany. Her research interest includes formal methods, intelligent systems, machine learning, meta-heuristic and hyper-heuristic techniques. Since 2019, she has been actively involved in activities to introduce Computational Thinking to students and teachers in elementary and middle schools in Bandung, Indonesia. She was once the head of the Bebras Bureau of Parahyangan Catholic University. She is also the editor of an Informatics book, a textbook for students and teachers of Informatics subjects in grades VII and X, which is published by the Indonesian Ministry of Education, Culture, Research, and Technology.

Latent and Evident Knowledge to Compose and to Solve Tasks in Informatics

Pavel S. PANKOV, Artem A. BELYAEV

Institute of Mathematics, Kyrgyzstan

Kyrgyz-Russian Slavic University, Kyrgyzstan

e-mail: pps5050@mail.ru, artem_belyaev@mail.ru

Abstract. On the base of the IOI Syllabus, the following problem was stated: what additional mathematical knowledge (including one invented by jury before OI and by contestants during OI) and other branches of knowledge are permissible in tasks? Socrates' idea (a human has vast latent knowledge which can be extracted by corresponding quests) is involved. Mathematical knowledge, special (physics, chemistry, geography) knowledge, general, or common knowledge are considered. It is demonstrated that the well-known notion "Turing-complete language" does not include time and cannot be called computationally universal. The Time dependent tasks and the corresponding time checker are proposed.

Keywords: Informatics, Olympiad, latent knowledge, special knowledge, time task.

1. Introduction

The International Olympiad in Informatics Syllabus (2022) (<https://ioinformatics.org/page/syllabus/12>) provides all necessary knowledge ("included topics") in mathematics to be involved in Olympiad tasks and "forbidden to be necessary" knowledge ("excluded topics").

Consider arising problems on example of the ancient.

Task 1 "Heads-feet". Given natural numbers H, F in $2 \dots 1000$. Geese and cats together have H heads and F feet. How many geese and how many cats are there? If it is impossible, output 0 0.

Firstly, the task must be "culturally neutral". For instance, "Geese and platypuses" belong to Australian culture (although "wombats" were at IOI'2013).

"Included topics" are not enough to solve this task. The contestant is to use "latent knowledge" not related to mathematics and informatics. It is also stipulated in Syllabus (2022): *Contestants are not expected to have knowledge of these topics. ... However ... The ISC may wish to include such a competition task in order to broaden the scope of the IOI... the ISC will make sure that the task can reasonably be solved without prior knowl-*

edge of the particular topic, and that the task can be stated in terms of $\sqrt{\dots}$ concepts in a precise, concise, and clear way.

If we add an explanation: *a goose has one head and two feet; a cat has one head and four feet* then the task loses sense: arbitrary numbers can be substituted (*fabulous geese and cats...* is used to be added; we proposed “naturalness” (Pankov, 2008) to avoid such technique). Also, such addition violates the demand “short and elegant formulations” (Dagienė *et al.*, 2007).

If we add, as usually, strict mathematical formulation of task: *find such natural numbers G and C that $G + C = H$ and $2 \cdot G + 4 \cdot C = F$* then, firstly, it becomes almost a solution, and secondly, Linear algebra is an “explicitly excluded topic” in Syllabus (2022).

Meanwhile, if this task is given without such addition and the contestant (or their coach) protests: *Linear algebra is an “explicitly excluded topic” in Syllabus (2022) that is why I could not solve this task* then the jury may respond: *Neither linear equations nor other notions of Linear algebra are mentioned in the task. If you reduce the task to a system of linear equations then it’s your problem. The task can be solved, for instance, by brute force method.*

We will consider the tasks without redundant mathematical formulation; “thin”, “light”, “little”, ratio of car length to road length ... mean “neglectable”.

The general problem arises: What “latent knowledge” not related to mathematics and informatics is permitted?

Besides of “included topics” and “explicitly excluded topics” listed in Syllabus (2022), there are unbounded mathematical topics including ones invented by the authors of tasks and the jury before the OI and by contestants during the OI (Pankov *et al.*, 2015; Pankov *et al.*, 2018). What mathematical topics can be meant latently?

In Section 2 we discuss various mathematical topics including such “discoverable ones”.

We proposed wider use of the achievements of science and technologies to create tasks (Pankov, 2010). Section 3 discusses “special knowledge” being necessary to solve such tasks.

Section 4 reviews the “latent general knowledge” of life and imagery ability for successfully solving of the tasks.

In Section 5 we propose a new type of tasks on measuring real time, discuss and improve “universal algorithmic language”.

Remark. Issues under discussion in this article cannot be covered by any general definitions or explanations. They can be demonstrated by examples only.

2. Mathematical Knowledge

The following tasks use topics which are not mentioned or are explicitly excluded from Syllabus (2022). Nevertheless, they can be posted and be solved successfully.

A popular at initial stages of OIs and of mathematical Olympiads (with a fixed large N)

Task 2 “Last digit”. Given integer number N in $2 \dots 10^{12}$, what is $F(N) := (\text{last digit of } 7^N \text{ in decimal system})$?

Most of contestants will be successful in such a task.

If the contestant (or their coach) protests: “Fermat’s little theorem is not included in Syllabus (2022) that is why I could not solve this task” then the jury may respond: “You could count $F(2)$, $F(3)$, $F(4)$, $F(5)$... and note a regularity”. That is, the ability to make small mathematical discoveries in addition to knowledge within Syllabus (2022) is meant at OIs.

Geometry in 3D is explicitly excluded in (Syllabus, 2022). But

Task 3 “Null-transportation”. There are three distinct $10^6 \times 10^6$ -squares A , B , C on a plane, crossing a boundary of a square is prohibited. Each square has its own coordinate system $0 \leq x, y \leq 10^6$. All numbers are integer. The cost of passing from a point $A(i, j)$ to points $A(i + 1, j)$ or $A(i - 1, j)$ or $A(i, j + 1)$ or $A(i, j - 1)$ (if they belong to the square), as well as for squares B and C , is 2 Euros. The cost of null-transportations $A(i, j) \leftrightarrow B(i, j)$ or $C(i, j) \leftrightarrow B(i, j)$ is 10 Euros. Given numbers x, y, u, v in $0 \dots 10^6$, find the minimal cost of passing from the point $A(x, y)$ to the point $C(u, v)$.

Obviously, any way of “common” search in the graph with $3 \cdot (10^6 + 1)^2$ vertices is hopeless. However, most of contestants will be successful in this task.

Remark. If the contestant protests: “3D-Manhattan metrics is explicitly excluded in Syllabus (2022) that is why I could not solve this task” then the jury may respond: “Neither a 3D-space nor Manhattan metrics are mentioned in the text of task. If you interpret the media as a 3D-space then it is your problem”. If the contestant protests: “There are not algorithms to find an optimal way in such a vast graph during one second” then the jury may respond: “If you interpret the media as a graph then it’s your problem”.

Euler’s formula for planar graphs is not mentioned in Syllabus (2022).

Task 4 “Triangles”. Let us call an intersection (point only) of two drawn segments as “node”, the drawn segment between two nodes without inner nodes as “fragment” and a drawn triangle without drawn inner points as “domain”. Let a triangle UVW be drawn, number N of nodes is 3, number F of fragments is 3, number D of domains is 1. The operation “draw a segment splitting any domain into two domains” was executed some times. Given numbers N and D in $10 \dots 10^6$, $N - 2 \leq D \leq 2N - 6$, find number F . If it is impossible, output 0.

Example 4.1. 1st operation: draw segment UP , point P is in VW ; 2nd operation: draw segment VQ , point Q is in UP ; 3rd operation: draw segment WQ . The result is $N = 5$; $F = 8$; $D = 4$.

At first, contestants will be frightened by arbitrariness but after some attempts most of them will be successful in this task.

Remark. If the contestant’s coach protests: “Euler’s formula for planar graphs is not an included topic in Syllabus (2022)” then the jury may respond: “There are other ways to solve the task, for instance, constructing a sequence of operations yielding given N and D .”

For convenience, we will use the denotation $\lfloor \cdot \rfloor$ also for rounding down to integer.

“Calculus” is explicitly excluded in Syllabus (2022).

Nevertheless, we (Pankov, 2013) proposed tasks of type.

Task 5 “Minimization”. *Given a natural number F in $1 \dots 10^300$; find such natural number X that the expression $H(X) := X^3 + F / (X + 1)$ is minimal; if there are some such numbers then output the greatest of them.*

The author of the task is obliged to calculate the derivative $H'(X)$ and to prove that it increases for (real) $X > 0$. But the contestant who does not know the notion “derivative” can easily guess (feel) that $H(X)$ is a unimodal function, guess and implement the following effective algorithm. $H(0) = F + 1$, $H(X) > X^3$. Hence, it is enough to consider X between $L = 0$ and $M = 10^400$.

Algorithm: Repeat {Let $P := \lfloor (3*L + M)/4 \rfloor$; $Q := \lfloor (L + 3*M)/4 \rfloor$; if $H(P) > H(Q)$ then $L := P$ else $M := Q$ } until $M - L \leq 5$. Calculate and compare $H(L)$, ..., $H(M)$.

“Non-trivial calculations on floating point numbers, manipulating precision errors; trigonometric functions” are explicitly excluded in Syllabus (2022).

Some countries, organizations, companies, firms (alleged sponsors) have central-symmetric elements on their coat-of-arms, logos.

Corresponding modifications of the following task may be in their honor (Pankov *et al.*, 2009).

Task 6 “Regular polygon”. *Given integer numbers N in $2..2023$, K in $1..64$. The center of the regular 64-polygon is in $(0; 0)$, the 1st vertex is in $(N; 0)$, vertices are numbered counterclockwise. Find such integers X and Y that the K th vertex is within the square $(X, X + 2) \times (Y, Y + 2)$. If there exist some such pairs output one of them.*

Remark. The task to find a square $[X, X + 1) \times [Y, Y + 1)$ is close to unresolvable one due to the following theorem of constructive mathematics: the problem of distinguishing a computable real number from zero is unsolvable.

The author’s solution of the task is a little rational-numbers-interval-analysis-soft. Trigonometric functions are not used in the solution; coordinates of vertices are calculated by formulas for vectors, such as $V[1] := (N; 0)$; $V[17] := (0; N)$; $V[9] := (V[1] + V[17]) / |(V[1] + V[17])| * N \dots$ (Euclidean distances, Pythagorean theorem are included in Syllabus (2022)).

But such solution takes about half of hour to type even for an experienced contestant. And the contestant writes a program for floating point numbers $X1, Y1$ in a minute:

Let $X1 := N * \cos(2.*\pi/64.*(K - 1))$; $Y1 := N*\sin(2.*\pi/64.*(K - 1))$;

$X := \text{floor}(X1)$; If $X1 - X < 0.5$ then $X := X - 1$; $Y := \text{floor}(Y1)$; If $Y1 - Y < 0.5$ then $Y := Y - 1$;

Output (X, Y) .

Because of high accuracy of the built-in functions *cos* and *sin* such program should pass all tests successfully.

Will the jury accept this solution? Checking of texts of programs is permitted only to detect (Cheating, 2022): *contestants must not attempt to submit illegal programs as*

discussed above [not perform explicit input and output operations...] ... to gain access ...to store information...to access any machine ... to reboot...

3. Special Knowledge

In our opinion, the tasks based on real facts and scientific laws are useful for the contestants because they are natural, they demonstrate the diversity of the world and prepare for a future fruitful activity.

Consider some branches of knowledge.

3.1. Physics

Velocity is a common physical notion. Nevertheless, by our experience, even simple tasks with piecewise-uniform speed are very difficult for contestants (see Section 5 below for probable reason of it).

Task 7 “Villages”. (Kyrgyzstan, Republican OI, IV stage, 2022). There are five villages on a straight road. The car started from the beginning of the road at 5.00 am. From 6.00 am to 7.00 am the cattle go to the pasture along villages, so the car speed within villages is 6 km/h, otherwise the car speed is 60 km/h. Find the coordinates (in meters) of the car in M minutes after 5.00 am.

Input: $M < 216$; coordinates of villages (km) $0 < B1 < A1 < B2 < A2 < B3 < A3 < B4 < A4 < B5 < A5 < 216$.

Gravitation, the lever law (by our opinion, it is natural and can be felt or guessed from the example).

Task 8 “Weighing” (Kyrgyzstan, National OI, I stage, 2023). Points ... $-3, -2, -1, 0, 1, 2, \dots$ are marked on a long ruler at equal distances between them. The ruler is suspended in the middle, at the point 0. There is plenty of a flour, a light bag for flour, and weights of natural numbers (given) P kg and (given) Q kg. A bag of flour and weights can be hung on a ruler, only at marked points. There cannot be two loads at same point. It is required to weigh (given) Z kg of flour.

Find the minimum possible length of the segment occupied by the weights. Example 8.1. (with comments): $P = 2$ (hung at “1”), $Q = 50$ (hung at “2”), $Z = 102$ (hung at “-1”) $\rightarrow 3$.

Task 9 “Non-symmetrical scale” (Kyrgyzstan, Republican OI, III stage, 2023). Given natural numbers P, Q, Z in $1 \dots 2023$. The (very light) bowls of the scale are suspended at distances P and Q (horizontally) from the suspension point of the scale. To weigh Z kg how many 1-kg-kettlebells are necessary? Example 9.1. $P = 20, Q = 10, Z = 1993 \rightarrow 998$.

Gravitation, properties of liquid.

Task 10 “Rain”. (Kyrgyzstan, Republican OI, II stage, 2023). Given integer number K in $4 \dots 6$. K thin rectangular walls were built on the plane in “north-south” or “east-west” directions (two walls can be intersected). It is raining, raining... How much water do the walls hold? (“zero” can also be).

Input. The first row contains K . The next K rows contain five natural numbers $X1, Y1, X2, Y2, Z$ in $1 \dots 10$, the coordinates of the end-points and height of the i -th wall, $i = 1 \dots K$.

Properties of snow (snow is not “solid” in physical terminology; it is a unique object).

Task 11 “Snow” (Kyrgyzstan, Republican OI, III stage, 2006). Let the streets in the city [Bishkek] form a rectangular grid (with coordinates), all blocks are 1×1 . The firm Logic [sponsor] is located at a given crossing (X, Y) . Two friends wish to come to Logic. Now the first is at the crossing $(X1, Y1)$, the second is at the crossing $(X2, Y2)$. Because of the plentiful snowing they wish to minimize the trampled path (the sum of paths trampled by the first, by the second and by the both going together). Write a program calculating the minimal length of path.

Law of reflection.

Task 12 “Mirrors”. Given integer numbers P and Q in $1 \dots 10^6$. The rectangle $ABCD$ is made of four mirror segments, $|AB| = |CD| = P$, $|AC| = |BD| = Q$. The ray came out of the vertex A along the bisector of the angle BAC . What vertex will the ray come in? Example 12.1.: 6000 1500 $\rightarrow B$

If the contestant is doubt in their memory or guessing on the law then the example confirms it.

Law of impulse conservation.

Task 13 “Carts”. (Kyrgyzstan, Republican OI, III stage, 2023). All data are integer. The “zero” point was marked on a horizontal long straight road. Two small carts move without friction along the road. If they collide then they concatenate and then move together. Their weights $S1$ and $S2$ (kilograms), initial locations $J1 \neq J2$ (meters), and initial velocities $Y1$ and $Y2$ (meters/second) are given. Where will the first cart be after U (seconds)? Give the answer as a fraction.

Example 13.1. $U = 8, S1 = 7, S2 = 7, J1 = 50, J2 = 51, Y1 = 1, Y2 = 0 \rightarrow 9/2$

X-rays, tomography.

General task 14 “Restoration”. To restore an object (an image) by its projections (by sums along columns, rows).

3.2. Chemistry

Task 15 “Equalization”. Chemical elements are denoted by an uppercase letter or by an uppercase letter and a lowercase letter; number of such atoms in the mole-

cule (less than 100), if it is greater than one, is written after. Given molecules $M1$, $M2$, $M3$. Find such a non-negative (the least possible) integer numbers $N1$, $N2$, $N3$ that $N1 \cdot M1 + N2 \cdot M2 = N3 \cdot M3$. If it is impossible, output 0 0 0.

Example 15.1. $H2\ O2\ H2O \rightarrow 2\ 1\ 2$. Example 2. $Bb17Bb3\ Sx2\ Bb22 \rightarrow 11\ 0\ 10$.

Remark. “Chemical language for molecules” is non-formal, writings may be more or less detailed. To avoid questions, “Bb” is written twice in one molecule and “Sx” is not used.

3.3. Geography, Astronomy

Task 16 “Above the equator”. All numbers are integer. Western longitude is denoted with “-“. Given (N in 2..360) distinct points with latitudes (in $-179..180$ degrees) on the equator. A geostationary satellite can cover (K in 1..30) degrees on the equator. How many geostationary satellites are necessary to cover all points? Example 16.1. $K = 2$, $N = 4 / 50\ -179\ 179\ -45 \rightarrow 3$.

Task 17 “Satellite”. Angles are measured in degrees. All numbers (except V) are integer. There is a planet P (its center C) with a satellite (S) far from the Earth (E). The straight-line EC is in the plane of the circular orbit of S . Denote the angle between EC and CS as A (if S is to the left of EC then let “ $-A$ ” be written). When S is before or behind P ($|A| < K$), S is not seen. When S is seen, we can measure A but we cannot detect whether S is farer or nearer than C . Suppose that the angular velocity of S be (rational number) V /hour.

By results of series of N observations on A at 0, 1, 2, ... $N - 1$ hours find the minimal value of V .

Input: numbers K in 2..45, N in 2..5 / N numbers in $-90..90$ (“0” means “not seen”).

Example 17.1. $30\ 2 / -84\ -84 \rightarrow 12/1$. Example 17.2. $30\ 2 / 84\ 86 \rightarrow 2/1$.

Example 17.3. $30\ 3 / 31\ 0\ -32 \rightarrow 63/2$.

3.4. Linguistics

Similar tasks for Kyrgyz language were used to be given. To be “culturally neutral”:

General Task 18 “Ciphering”. A number in 1..99 (without the sign “-“) was written in lowercase letters. Each letter was changed (bijective) to an uppercase letter. One of the letters was erased (or: One of the letters was changed; or Any letter was added ...). Restore the greatest possible value of the number.

Example 18.1. $WZ \rightarrow 10$. Example 18.2. $QWQZSKKG \rightarrow 59$.

4. General, or Common Terminology and Knowledge

Notions of language, odometer are mentioned in Syllabus (2022) as examples.

Traditional notions are persons (IOI participants, players, travelers), animals (it is better that their properties in the task be similar to their actual ones, for instance, jumping frog at IOI'2002; hounds and moths with GPS; worms as mathematical tasks characters from ancient times), ways, roads, crosses, villages (as points or segments on highways) and cities (large, with rectangular street grid), walls (as segments on a plane), cars (as points), pixels ...

We proposed voxels, timexels, 2D-printers, 3D-printers, future (3D + T)-printers for wider using (Pankov *et al.*, 2021).

Various natural tasks can be composed on the notions of (ideal, thin, inextensible) rope and other idealized household items.

Task 19 “Rope”. *Given a rope of length L and a graph, all its arcs have lengths 1 and are thin tubes. Rope is arranged within arcs, without touching itself. Can Rope be drawn through all arcs without self-touching? If it is possible, find the minimal time to do it.*

Every language carries its own specific notions. Examples:

Latin “formula”, “registration”, “optimization” and many others; Ancient Greek “program”, “axiom” and many others. It is difficult to explain them but they became international and can be used.

Fijian “taboo” (and its versions in other Pacific Ocean languages) is convenient to be used in some Olympiad tasks but it would not be “culturally neutral”.

Kyrgyz “irgöö” is more general than the word “synergetic” (Kenenbaeva, 2014). It yields self-ordering in stochastic processes. As statistics is an “explicitly excluded topic” in Syllabus (2022), such process can be imitated by “arbitrary operations, operations in arbitrary order”, as in Task 4. Will such task be “culturally neutral”?

The next layer is “relations”: neighbors, friends, teacher and students, teammates...

Many verbs mean various facts, laws, operations: glue, cut, paint (are used for graphs), pour (see Task 10), fold ...

Task 20 “Folding”. *Given a number N and a (large) polygon glued of equal 1×1 -squares. We can fold it by lines of gluing. How many foldings are necessary to obtain a (multi-layer) polygon with area not greater than N ?*

Every native speaker has vast knowledge about and some skills. What of it can be used in tasks?

For example, is latent knowledge in Task 1 related to common one or to zoological one?

Are properties of water (liquid) in Task 10 physical or obvious?

Properties of snow cannot be explained formally, mathematically. By what knowledge do contestants solve Task 11?

Obviously, for successful solving of Task 10 the contestant is to have the property of measuring imagery (Pankov, 1996).

5. On Universal Algorithmic Languages, Time Tasks and Random Phenomena

Nowadays computers cover anyway almost all universe known to the mankind. In our turn, we are to cover more aspects of life by Olympiad tasks. Firstly, what algorithmic language is suitable for such purposes?

Well-known statements which are also meant in OI:

“a system of data-manipulation rules (such as a computer’s instruction set, a programming language, or a cellular automaton) is said to be Turing-complete or computationally universal if it can be used to simulate any Turing machine” (https://en.wikipedia.org/wiki/Turing_completeness).

But by our opinion, such language is neither “complete” nor “universal”. A simple command “in (2 ± 0.01) second output 2023” or a natural condition “if $(1.99 < \text{time} < 2.01)$ and $(5.01 < X < 5.02)$ then ...” cannot be written in such language.

Probably, “time” was not included in “universal language” because a human does not have a built-in timer unlike some animals. But various automats and devices since XIX century had. It turns out that developers of this important notion in middle of XX century did not pay attention to them.

To broaden the horizons of participants we propose to include the following type of tasks in initial stages of OI:

Task 21 “Just-in-time”

Given natural numbers A , B and N . Write a program to output $(A + B)$ in $(N \pm 10)$ milliseconds.

Input: A , B in $-10^9 \dots 10^9$, N in $100 \dots 1000$.

Output: Sum $(A + B)$ and an integer number denoting actual time from running the program (milliseconds)

Example 21.1. Input: 5 6 900 Output: 11 905 or 11 897 or ...

The full text of the task is at <https://cloud.mail.ru/public/S9wJ/dqzHxWPNa>

The checker is at <https://cloud.mail.ru/public/DTgK/QFx8mrP7K>

A text of program is at <https://cloud.mail.ru/public/gn17/hpnoEJWjG>

This task is also included in:

<https://olymp.krsu.edu.kg/GeneralProblemset.aspx>

Remark. To support the idea of Turing-complete language, the physical Church-Turing thesis was proposed, for instance, (Arrighi *et al.*, 2012): any function that can be computed by a physical system can be computed by a Turing Machine.

The following physical system (Pankov *et al.*, 2018a) refutes this thesis.

The surface (made of tin) consists of six triangles with the following seven vertices (in cylindrical coordinates: radial distance; angle; height): $A[0] := (0''; 0; 0'')$; $A[K] := (15''; \pi * i/3; (5 + 2 * (-1)^K''))$; $K = 1 \dots 6$.

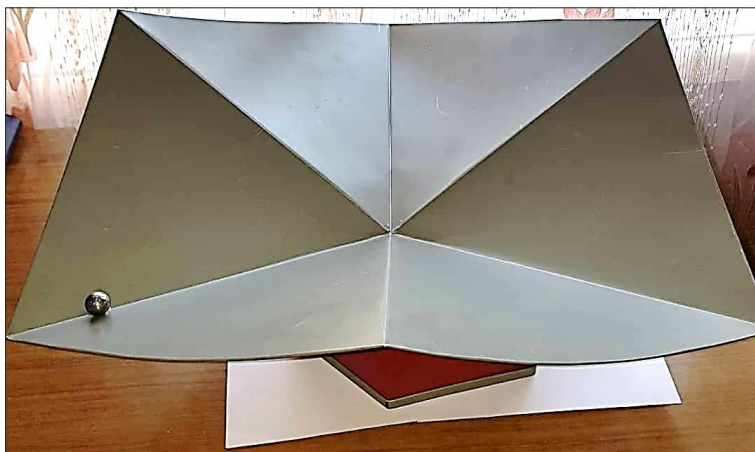


Fig. 1. Pavel Pankov, Sabina Tagaeva © 2018 Made by Nuraly Niyasbekov.

The little ball (made of steel) is launched from the point $A[1]$. It rolls down along the “valley” to the point $A[0]$, rolls up a bit along the “ridge” $A[0] - A[4]$, falls off left to “valley” $A[5] - A[0]$ or right to “valley” $A[3] - A[0]$ randomly, rolls down to the point $A[0]$, rolls up a bit along the “ridge” $A[0] - A[2]$ or along the “ridge” $A[0] - A[6]$ respectively etc. Actually, the ball rolls up twice before stop in the point $A[0]$ (for instance, in 4 second). Each launch initiates another function $[0, 4] \rightarrow \mathbb{R}^3$ and none Turing Machine can calculate it.

6. Conclusion

We hope that this paper would draw attention to more general problem (especially for international students): what latent and evident knowledge and skills are necessary for successful life, study and work in our changing world?

Results from the focus group discussions showed that international students face challenges in their everyday life, dormitory life, campus life, social life and academic life. (Gebre et al., 2020)

Eynullayeva *et al.* (2021) examined whether the cultural adaptation levels of international students vary according to gender, place of residence, academic achievement level, education level, faculty they attend, and their age.

And how can OIs (covering millions young people at initial stages) contribute to this problem?

References

- Arrighi P., Dowek G. (2012). The physical Church-Turing thesis and the principles of quantum theory. *International Journal of Foundations of Computer Science*, 23(5), 1131–1145.
- Cheating (2022). <https://ioi2022.id/competition-rules/>
- Dagienė, V., Skupienė, J. (2007). Contests in programming: quarter century of Lithuanian experience. *Olympiads in Informatics: Country Experiences and Developments*, 1, 37–49.
- Eynullayeva, K., Gökalp, M., Hatunoglu, B.Y. (2021). Investigation of the Turkish Cultural Adaptation of International Students Living in Turkey. *European Educational Researcher*, 2(2), 155–169.
- Gebru, M. S., Yuksel-Kaptanoglu, I. (2020). Adaptation Challenges for International Students in Turkey. *Open Journal of Social Sciences*, 8, 262–278.
- Kenenbaeva G. (2014). Framework Definitions of Effects and Phenomena and Examples in Differential and Difference Equations. *Journal of Mathematics and System Science*, 4, 766–768.
- Pankov P.S. (1996). Independent learning for open society. *Collection of papers as results of seminars conducted within the frames of the program "High Education Support"*. Foundation «Soros-Kyrgyzstan», Bishkek, issue 3, 27–38.
- Pankov, P.S. (2008). Naturalness in Tasks for Olympiads in Informatics. *Olympiads in Informatics: Country Experiences and Developments*, 2, 16–23.
- Pankov, P.S. (2010). Real Processes as Sources for Tasks in Informatics. *Olympiads in Informatics*, 4, 95–103.
- Pankov, P.S. (2013). Tasks in Informatics of Continuous Content. *Olympiads in Informatics*, 7, 101–112.
- Pankov, P.S., Baryshnikov, K.A. (2009). Representational Means for Tasks in Informatics. *Olympiads in Informatics*, 3, 101–111.
- Pankov, P.S., Imanaliev, T.M., Kenzhaliev, A.A. (2021). Automatic Makers as a Source for Olympiad Tasks. *Olympiads in Informatics*, 15, 75–82.
- Pankov, P., Janaliev, J., Naimanova, A. (2015). Inductive and experimental studying of mathematical subjects (mathematical facts and notions which can be discovered independently), *LAP Lambert Academic Publishing*, Saarbrücken.
- Pankov, P.S., Kenzhaliev, A.A. (2018). Combinatorial property of sets of boxes in Euclidean spaces and theorems in Olympiad tasks. *Olympiads in Informatics*, 12, 111–117.
- Pankov, P.S., Tagaeva, S.B. (2018a). Computer and real simulation of phenomenon of strange attractor by system of differential equations. *Herald of Institute of Mathematics of National Academy of Sciences of Kyrgyz Republic*, 1, 17–23.
- Syllabus (2022). *The International Olympiad in Informatics Syllabus*.
<https://ioinformatics.org/page/syllabus/12>



P.S. Pankov (1950), doctor of physics-mathematics sciences, prof., corr. member of Kyrgyzstani National Academy of Sciences (KR NAS), was the chairman of jury of Bishkek City OIs, 1985–2013, of Republican OIs, 1987–2012, participates in National Olympiads since 2020, was the leader of Kyrgyzstani teams at IOIs, 2002–2013, 2018–2022. Graduated from the Kyrgyz State University in 1969, is a head of laboratory of Institute of mathematics of KR NAS.



A.A. Belyaev (1978), Kyrgyz-Russian Slavic University. Deputy leader at IOI'2018. Regional Director of NERC ICPC Kyrgyzstan Regionals.

Understanding and Designing Recursive Functions via Syntactic Rewriting

Tom VERHOEFF

*Mathematics and Computer Science, Eindhoven University of Technology
Groene Loper 5, 5612 AE, Eindhoven, Netherlands
e-mail: t.verhoeff@tue.nl*

Abstract. Recursion is considered a challenging programming technique by many students. There are two common approaches intended to help students understand recursion. One of them is based on the operational semantics of function execution involving a stack, where students trace the execution of a recursively defined function for some concrete arguments. The other approach is based on the axiomatic semantics involving inductive reasoning with the contract of the recursively defined function. The former approach is not so helpful when designing recursive functions, whereas the latter can be helpful (being a special case of divide and conquer) but contracts can be hard to discover.

In this article, I will show a third approach. It is based neither on an operational nor an axiomatic semantics. Rather, it involves a rewriting semantics using program transformation by substitution, thereby inlining function calls. We show that this approach not only may help understanding, but can also be used to design recursive functions.

Keywords: computer science, education, programming, recursion.

1. Introduction

I have written about recursion before (Verhoeff, 2018, 2021) and until recently I would not have thought to have something significant to add. But while preparing a programming Q&A session for first-year mathematics students, I was struck by a (for me) new idea, which is the topic of this article.

Verhoeff (2018) presents the two common approaches to understand recursion. As example, consider the following Python code for the recursively defined function `print_bit_strings` (we left it undocumented for now on purpose):

```
1 def print_bit_strings(n: int, s: str = "") -> None:
2     if n == 0:
3         print(s)
4     else:
```

```

5     for b in "01":
6         print_bit_strings(n - 1, s + b)

```

The call `print_bit_strings(2)` prints

```

00
01
10
11

```

One could wonder why there is a need for this extra parameter `s`, and why the recursive call on line 6 has argument `s + b` rather than `b + s`.

To understand this, students could use the *operational* approach, where they trace the execution of a specific call of the recursive function in question through multiple levels of subsequent recursive calls. Beginning programmers perceive this execution as magical and confusing, because there is a single definition of the given recursive function, but multiple calls are simultaneously executing that same piece of code independent of each other. That is, each invocation can be at a different location in that code, and the parameters and other local variables can have different values. The execution of a recursive function traverses an imaginary *dynamic call tree* (Verhoeff, 2018, §3.1), where each node corresponds to the execution of a call, which then gives rise to zero or more subsequent recursive calls. The active invocations form a *root path* in this tree, and the ‘instruction pointer’ and values of local variables are stored on a *stack*, that grows and shrinks as the root path.

For example, the call `print_bit_strings(2, "s")` gives rise to the call tree in Fig. 1. This approach helps in understanding how a stack machine can correctly execute a recursively defined function in an imperative programming language. But in my experience it is neither helpful for reasoning about (the correctness) of recursive function definitions nor for designing them.

Alternatively, there is the *axiomatic* approach (Verhoeff, 2018, §4), which requires a specification of the recursive function in terms of a *contract* consisting of a *precondition* and a *postcondition*, such that

- if the precondition is satisfied *before* the function call,
- then the postcondition is satisfied *after* the function call.

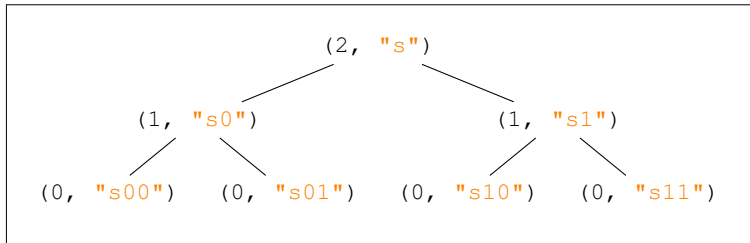


Fig. 1. Call tree for `print_bit_strings(2, "s")`, only showing parameters.

The docstring for `print_bit_strings(n, s)` could read:

```

7      """Print s + t for all strings t over "01" of length n,
8      in lexicographic order.
9
10     Assumption: n >= 0
11     """

```

That is, its precondition is $n \geq 0$ and its postcondition is: ‘for all strings t over “01” of length n , strings $s + t$ have been printed’. To reason about the call with parameters (n, s) , we assume as *induction hypothesis* that calls with parameters $(n_, s_)$ where $n_ < n$ work as specified by the contract. The design of function `print_bit_strings` can be argued as follows. The goal is to prove that lines 2–6 satisfy the contract, under the assumption that the recursive call on line 6 satisfies its contract; that is, it satisfies ‘if $n > 0$, then for all strings u of length $n - 1$ over “01”, it prints strings $s + b + u$ ’.

- If $n == 0$, then there is only one string t of length n , viz. the empty string. Observe that s extended with the empty string equals s (line 3). Thus, the contract is fulfilled by printing just s .
- If $n > 0$ then extensions of length n over “01” start with a single bit, say b , in “01”, followed by $n - 1$ more bits over “01”. Thus, by the induction hypothesis, the loop on lines 5–6 prints all required strings.

In this reasoning style, the induction hypothesis is sometimes referred to as the *recursive leap of faith* (Roberts, 1986; Rubio, 2018). I find this terminology unfortunate, because faith has nothing to do with it. There is already good terminology, viz. *induction hypothesis*. Trusting the compiler, runtime system, and hardware to execute recursive definitions faithfully, could be called a leap of faith. Students need to understand this implementation only once, e.g., by tracing an execution through the call tree and observing the role of the stack. But this is not needed (nor helpful) to understand specific recursive definitions, and certainly not for designing them.

2. Syntactic Rewriting

There is a third kind of semantics for programming languages, viz. based on *rewriting*. It is typically used to describe the semantics of more advanced language constructs in terms of simpler language constructs. For example, $i += 1$ can be rewritten into $i = i + 1$. That way, the meaning of $+=$ is defined, without the need to speak of how $+=$ works operationally, nor how to reason about it axiomatically. This rewriting is a purely syntactic operation and is also called a *semantics-preserving program transformation*. One sometimes calls the notation $+=$ *syntactic sugar*, because it does not make the language more expressive. It is a mere abbreviation that can be eliminated by rewriting, also known as *syntactic desugaring*.

2.1. Rewriting Function Calls, by Inlining

In the absence of recursion, the function mechanism is syntactic sugar for abbreviations that can be eliminated by a program transformation. Consider a void¹ function definition without **return** statements of the form

```
13 def func(param_1, param_2, ...):
14     body # containing param1, param2, ..., but no return
```

The call `func(expr_1, expr_2, ...)` can be eliminated as follows.

1. Replace the call `func(expr_1, expr_2, ...)` by

```
15     param_1, param_2, ... = expr_1, expr_2, ...
16     body
```

where `param_1, param_2, ...` are local variables.

2. Systematically rename any local variables in `body` whose name clashes with a name occurring in the context of the call.

When the expressions `expr_i` are free of *side effects*, also known as *referential transparency* (which will be the case in our examples), Step 1 can be replaced by a *double substitution*:

- 1.a. Replace the call `func(expr_1, expr_2, ...)` by `body`,
- 1.b. in which all occurrences of parameters `param_i` are simultaneously replaced by the corresponding argument expressions `(expr_i)`. The parentheses are needed to guarantee the proper evaluation order.

The result of this transformation is a program that is semantically equivalent to the original program. As a *compiler optimization* technique and as a *code refactoring* technique, it is also known as *inlining*. In Lambda Calculus, the double substitution corresponds to β -reduction.

For example, consider the Python function definition

```
17 def f(x, y):
18     z = x * y
19     print(z)
```

Then we can rewrite as follows

```
20     f(z - 1, z + 1)
21     print(z)
22
23 # inline call: x, y = z - 1, z + 1
24
```

¹ In some programming languages void functions are known as procedures. In Python, their body does not contain **return** `expr`. For non-void functions and **return**, see Appendix A.

```

25     z1 = (z - 1) * (z + 1) # z1: fresh local variable
26     print(z1)
27     print(z)

```

Note that in this case the parentheses are really needed to preserve the evaluation order of operators. When they are not needed, we silently omit them.

Non-recursive function definitions can be completely eliminated by inlining all their calls. This may result in faster execution, at the expensive of a larger code foot print. Recursive function definitions cannot be completely eliminated this way. Well, they can, provided we allow *infinite* program texts. Conceptually, there is nothing wrong with an infinite program text. Termination of the recursion corresponds to guaranteeing that only a finite (though unbounded) part of that infinite program gets executed.

2.2. Rewriting *if*-statements, by Deleting Dead Branches

It turns out that for the examples in §3 we also need rewriting rules for *if*-statements with constant conditions. These are the two relevant rewrite rules:

```

28     if True:
29         statement_suite_1
30     else: # unreachable
31         statement_suite_2
32
33 # delete dead else-branch
34
35     statement_suite_1

```

and

```

36     if False: # unreachable
37         statement_suite_1
38     else:
39         statement_suite_2
40
41 # delete dead if-branch
42
43     statement_suite_2

```

2.3. Rewriting Expressions, by Constant Folding

The final rewrite step that we use in the examples below is that of evaluating an expression involving only constants. As a compiler optimization technique this is known as *constant folding*. We label such rewrites by `simplify`.

3. Example for Understanding Via Rewriting

Let's rewrite the call `print_bit_strings(2)`:

```

44     print_bit_strings(2)
45
46 # inline call: n, s = 2, ""
47
48     if 2 == 0:
49         print("")
50     else:
51         for b in "01":
52             print_bit_strings(2 - 1, "" + b)
53
54 # delete dead if-branch: 2 != 0 ; simplify: 2 - 1 == 1 ; "" + b == b
55
56     for b in "01":
57         print_bit_strings(1, b)
58
59 # inline call: n, s = 1, b; rename local variables
60
61     for b1 in "01":
62         # print_bit_strings(1, b1)
63         if 1 == 0:
64             print(b1)
65         else:
66             for b2 in "01":
67                 print_bit_strings(1 - 1, b1 + b2)
68
69 # delete dead if-branch: 1 != 0 ; simplify: 1 - 1 == 0
70
71     for b1 in "01":
72         for b2 in "01":
73             print_bit_strings(0, b1 + b2)
74
75 # inline call: n, s = 10, b1 + b2; rename local variable
76
77     for b1 in "01":
78         for b2 in "01":
79             # print_bit_strings(0, b1 + b2)
80             if 0 == 0:
81                 print(b1 + b2)
82             else:
83                 for b3 in "01":

```

```

84         print_bit_strings(0 - 1, b1 + b2 + b3)
85
86 # delete dead else-branch: 0 == 0
87
88     for b1 in "01":
89         for b2 in "01":
90             print(b1 + b2)

```

So, by *equational reasoning*, the recursive function applied to argument 2 is equivalent to 2 nested `for`-loops. It is now believable that for argument `n`, the call is equivalent to `n` nested `for`-loops, since each recursive call adds a level of nesting:

```

91 # print_bit_strings(n)
92 for b1 in "01":
93     for b2 in "01":
94         ...
95         for bn in "01":
96             print(b1 + b2 + ... + bn)

```

The role of (accumulation) parameter `s` can apparently be viewed as collecting the state of all enclosing `for`-loops. We see that recursion enables one to write programs with a variable number of nested `for`-loops. Without recursion this is often not possible in an imperative programming language. However, see §4 for a way of doing it in Python without recursion.

To my delight, the IntelliJ IDE can do the refactoring steps of inlining calls of recursive functions and deleting dead branches in `if`-statements, but only for Java. It can't do them for Python (neither can VS Code). It can inline nonrecursive Python function calls, but not for direct recursive functions. Surprisingly, if the recursive function is duplicated and turned into a pair of mutually recursive functions, the IDE will refactor their function calls properly.

Finally, it is important to note the difference between tracing the execution and rewriting the program as ways of understanding recursion. Execution tracing suffers from the exponential blow up in branching recursion, whereas syntactic program rewriting does not. The latter reasons about the (unexecuted) program as a whole.

4. Designing Recursive Function Definitions

The rewriting approach described in the previous section to help understand recursive function definitions can also be used to help design such functions. The steps are as follows.

1. Write down a non-recursive program that solves a particular instance of the problem.

2. Decide which part will be done in a single layer of the recursion, which part will be handled by recursion, and which part is handled by preceding layers. Focus on the perspective of that single layer.
3. Introduce appropriate parameters to feed in data that comes from the preceding layers, and modify and pass them on to the lower layers. In particular, there will also be a parameter for the problem size.
4. Decide on the base case(s).
5. Define the recursive function; in particular, introduce an **if**-statement to distinguish the base case(s) and the ‘general’ case that adds a single layer.

4.1. First Design Example

As an example, consider the problem of printing all bit strings of length n .

1. A straightforward non-recursive program for $n = 3$ consisting of 3 nested **for**-loops:

```

97     for b1 in "01":
98         for b2 in "01":
99             for bn in "01":
100                 print(b1 + b2 + b3)

```

2. A single layer of the recursion does one **for**-loop, say with control variable b_2 , the loops nested inside will be handled by the recursive call, and the outer loops were done in the preceding layers:

```

101     # print_bit_strings(3)
102     #####
103     for b1 in "01": # preceding layers
104         #####
105         for b2 in "01": # < this will be done in one layer
106             #####
107             for b3 in "01": # handled by
108                 print(b1 + b2 + b3) # recursion
109             #####

```

Note that the part handled by recursion concerns a *generalization*, because those **for**-loops do not just contain `print(b3)`. Rather, they contain `print(b1 + b2 + b3)`. So, the generalization is that the recursive call must print `s = b1 + b2` extended with `b3`. That first part must come in via an extra parameter, say `s`. This also means that the call itself will receive that parameter, but then its value will be `b1`, received from the preceding layers. For the top-level call, we can take `s = ""`, because it is the unit of string concatenation.

3. Thus, we obtain the following structure:

```

110     #####
111     for b1 in "01": # preceding layers
112     #####
113         # print_bit_strings(2, b1) # call being designed
114         for b2 in "01":
115             print_bit_strings(1, b1 + b2) # recursive call
116             # which should inline as
117             # for b3 in "01":
118                 #     print(b1 + b2 + b3)

```

4. In case `n == 0`, there are no `for`-loops, and only a `print` statement, which can be fed with the parameter `s`.

5. This leads to the following definition, which we have seen before:

```

119 def print_bit_strings(n: int, s: str = "") -> None:
120     """Print s + t for all strings t over "01" of length n,
121     in lexicographic order.
122
123     Assumption: n >= 0
124     """
125
126     if n == 0:
127         # t == ""
128         print(s)
129     else:
130         # n > 0, write t == b + u for b in "01"
131         for b in "01":
132             print_bit_strings(n - 1, s + b)

```

Note the default value `s = ""`, corresponding to an empty context of preceding recursive layers.

In this approach, one can use the IDE refactoring technique known as *extract function*. It will introduce appropriate parameters to feed in values that are must be supplied.

4.2. Second Design Example

Consider a binary tree of depth 2 (Fig. 2, left). How can it be used to create a binary tree of depth 3? One way is to copy the binary tree of depth 2 and combine the two instances with a fork on top (Fig. 2, middle). This is the view we followed in the preceding example. But one can also grow a binary fork on each of the leaves of the binary tree of depth 2 to get a binary tree of depth 3 (Fig. 2, right).

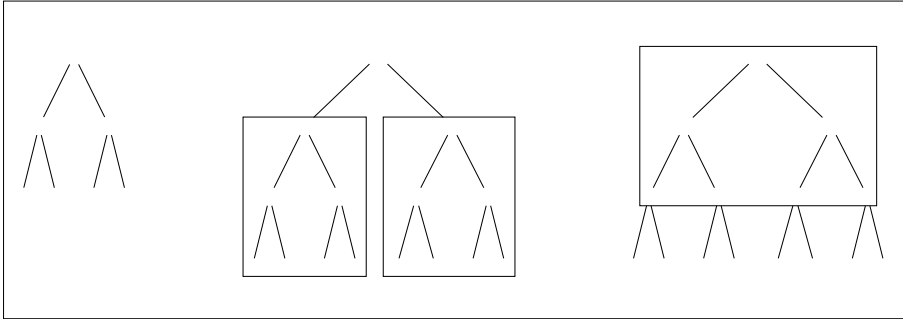


Fig. 2. Binary tree of depth 2 (left); two ways (middle, right) of constructing a binary tree of depth 3 from trees of depth 2.

Let's see how that alternative choice can be worked out in case of function `print_bit_strings`.

1. The non-recursive program is the same as above in §4.1.
2. A single layer of the recursion does one `for`-loop, but now the recursion will do the *outer* `for`-loops and the preceding layers did the *inner* loops:

```

133     # print_bit_strings(3)
134     #####
135     for b1 in "01": # handled by recursion
136         #####
137         for b2 in "01": # < this will be done in one layer
138             #####
139             for b3 in "01": # preceding
140                 print(b1 + b2 + b3) # layers
141             #####

```

This may look strange, but bear with me. Apparently, in the recursion, bit strings of length one shorter are produced, and these still need to be extended and printed. So, again, we see a *generalization*: rather than just print all bit strings, we need to apply a function to them, and this function is going to be an extra parameter, say `f`. The call we are designing receives this parameter, which represents the work to be done in the preceding layers (inner loops) For the top-level call, we can take `f = print`. The recursive call will receive a function (as parameter) that adds one `for`-loop around the given `f`.

3. Thus, we obtain the following structure:

```

142     # print_bit_strings(2, f)
143     def g(s: str) -> None:
144         for b2 in "01":
145             f(s + b2)
146

```



```

147     print_bit_strings(1, g)
148     # which should eventually inline as
149     # for b1 in "01":
150     #     g(s + b1)
151
152     # where def f(s: str) -> None:
153     #####
154         for b3 in "01":     # preceding
155             print(s + b3)   # layers
156     #####

```

4. In case $n == 0$, there are no **for**-loops and f can just be applied to `""`.

5. This leads to the following recursive definition:

```

157 def print_bit_strings(n: int,
158                          f: Callable[[str], None] = print
159                          ) -> None:
160     """Apply f to each string over "01" of length n,
161     in lexicographic order.
162
163     Assumption: n >= 0
164     """
165     if n == 0: # only the empty string has length 0
166         f("")
167     else: # n > 0
168         def g(s: str) -> None:
169             for b in "01":
170                 f(s + b)
171
172         print_bit_strings(n - 1, g)

```

Let's now see if we can understand this recursive definition by rewriting the call `print_bit_strings(2)`:

```

173     print_bit_strings(2)
174
175 # inline call: n, f = 2, print
176
177     if 2 == 0:
178         print("")
179     else:
180         def g(s: str) -> None:
181             for b in "01":
182                 print(s + b)

```

```

183
184     print_bit_strings(2 - 1, g)
185
186 # delete dead if-branch: 2 != 0 ; simplify: 2 - 1 == 1
187
188     def g(s: str) -> None:
189         for b in "01":
190             print(s + b)
191
192     print_bit_strings(1, g)
193
194 # inline call: n, f = 1, g; rename new local function g -> g2
195
196     def g(s: str) -> None:
197         for b in "01":
198             print(s + b)
199
200     if 1 == 0:
201         g("")
202     else:
203         def g2(s: str) -> None: # renamed local function
204             for b in "01":
205                 g(s + b)
206
207         print_bit_strings(1 - 1, g2)
208
209 # delete dead if-branch: 1 != 0 ; simplify: 1 - 1 == 0
210
211     def g(s: str) -> None:
212         for b in "01":
213             print(s + b)
214
215     def g2(s: str) -> None:
216         for b in "01":
217             g(s + b)
218
219     print_bit_strings(0, g2)
220
221 # inline call g(s + b); rename control variables
222
223     def g2(s: str) -> None:
224         for b1 in "01": # renamed control variable
225             for b2 in "01": # renamed control variable
226                 print((s + b1) + b2)

```

```

227
228     print_bit_strings(0, g2)
229
230 # inline call: n, f = 0, g2; rename new local function g -> g3
231
232     def g2(s: str) -> None:
233         for b1 in "01":
234             for b2 in "01":
235                 print((s + b1) + b2)
236
237     if 0 == 0:
238         g2("")
239     else:
240         def g3(s: str) -> None:
241             for b in "01":
242                 g2(s + b)
243
244         print_bit_strings(0 - 1, g3)
245
246 # delete dead else-branch: 0 == 0
247
248     def g2(s: str) -> None:
249         for b1 in "01":
250             for b2 in "01":
251                 print((s + b1) + b2)
252
253     g2("")
254
255 # inline call g2("")
256
257     for b1 in "01":
258         for b2 in "01":
259             print("" + b1 + b2)
260
261 # simplify: "" + b1 = b1
262
263     for b1 in "01":
264         for b2 in "01":
265             print(b1 + b2)

```

Now, the extra parameter (*f*) accumulates the work to be done, and in the base case it is applied to the empty string. Such a parameter is called a *continuation*. Observe that this definition of `print_bit_strings` is *tail recursive* and thus can easily be transformed into a **while**-loop:

```

266 def print_bit_strings(n: int) -> None:
267     """Print all strings over "01" of length n,
268     i   n lexicographic order.
269
270     Assumption: n >= 0
271     """
272     f = print
273
274     while n > 0:
275         def g(s: str, f=f) -> None:
276             for b in "01":
277                 f(s + b)
278
279                 n, f = n - 1, g
280
281     f("")

```

Note that here `g` needs an extra parameter `f` with default value `f`, to ensure that the definition of `g` is a *closure* that properly captures the function object currently bound to the name `f` at the moment of definition. Without that `f` parameter, the definition of `g` would contain an ‘open’ (un-dereferenced) name `f`, which will be looked up during execution of `g` to find the value bound to `f` at the moment of execution, rather than at the moment of definition.

By redefining `f` directly, `print_bit_strings` can be simplified to

```

282     f = print
283
284     for _ in range(n):
285         def f(s: str, f=f) -> None:
286             for b in "01":
287                 f(s + b)
288
289     f("")

```

Note that `f` here is not defined recursively, since the `f` in its body is the parameter, which is bound to the earlier value of `f`.

Apparently, in Python one can write a non-recursive program that behaves like a variable number of nested loops. Actually, the program constructs a function that behaves like those nested loops. Thus, this is a form of *metaprogramming*.

5. Conclusion

I have described and illustrated how a rewriting semantics can help understand and design recursive function definitions. This approach complements the traditional approaches based on operational semantics (execution tracing) and axiomatic semantics (contracts). I am not claiming that the approach via rewriting is the best, but I do find it better than execution tracing, because (i) it is purely syntactic, (ii) does not need a stack to distinguish the states of concurrently active recursive calls, (iii) nor does it suffer from any exponential blow up. Furthermore, the rewriting approach may help in discovering and formulating generalized contracts, which are needed for the approach via axiomatic semantics.

There is a clear relationship to *functional programming*, whose semantics can be based on Lambda Calculus, which has a rewriting semantics via β -reduction (inlining). Note that a rewriting semantics does not need a stack. In this article, I have shown that this approach also can work for imperative programs. It thus allows *equational reasoning* on the level of whole programs. However, some care is needed, in particular when **return** statements are used and when expressions can have side effects, causing a lack of *referential transparency*.

The approach via rewriting would benefit from IDE support for the relevant *refactoring techniques*, because manual rewriting is tedious and error-prone. Unfortunately, such support is currently rather limited (JetBrains IntelliJ can do it for Java).

A word of warning is in place concerning the examples. The rewriting approach can help to come up with recursive definitions. But these definitions may still have performance issues. There are other techniques to improve the performance of recursively defined functions, e.g., see Verhoeff (2018). Appendix B offers better ways of printing bit strings in Python. To show the power of a purely functional programming language, I have included some Haskell programs for generating bit strings in Appendix C.

I hope also that I have shown some nifty uses of Python. Maybe the recent Python compiler named Codon (Shajii, 2023) can make Python interesting for use in programming contests such as the IOI.

Acknowledgment

I would like to thank Berry Schoenmakers and Sten Wessel (TU Eindhoven, Netherlands) and Radu Negulescu (Ontario, Canada) for helping me improve this article.

References

- Roberts, E. (1986). *Thinking Recursively* (1st Ed.). Wiley.
- Rubio-Sánchez, M. (2018). *Introduction to Recursive Programming*. Taylor & Francis.

- DOI: 10.1201/9781315120850
- Shajii, A. *et al.* (2023). Codon: A Compiler for High-Performance Pythonic Applications and DSLs. In: *Proceedings of the 32nd ACM SIGPLAN International Conference on Compiler Construction*. ACM, pp. 191–202. DOI: 10.1145/3578360.3580275
- Verhoeff, T. (2018). A Master Class on Recursion. In: *Adventures Between Lower Bounds and Higher Altitudes*. Lecture Notes in Computer Science Vol. 11011. Springer, pp. 610–633. DOI: 10.1007/978-3-319-98355-4_35
- Verhoeff, T. (2021). Look Ma, Backtracking without Recursion. (IOI Conference 2021). *Olympiads in Informatics*, 15, 119–132. DOI: 10.15388/ioi.2021.10
- Verhoeff, T. (2023). Git repository with source code for “Understanding and Designing Recursive Functions via Syntactic Rewriting”. (Accessed 29 April 2023)
<https://gitlab.tue.nl/t-verhoeff-software/code-for-understanding-recursion>



T. Verhoeff is Assistant Professor in Computer Science at Eindhoven University of Technology, where he works in the group Software Engineering & Technology. His research interests are support tools for verified software development and model driven engineering. He received the IOI Distinguished Service Award at IOI 2007 in Zagreb, Croatia, in particular for his role in setting up and maintaining a web archive of IOI-related material and facilities for communication in the IOI community, and in establishing, developing, chairing, and contributing to the IOI Scientific Committee from 1999 until 2007.

Appendix A. Python Example with Non-void Function

Eliminating the call of a non-void² function is a bit more involved than for void functions. To keep things simple, we will assume that the return statements in the function body occur at *tail positions*, that is, if the **return** statement would have been a function call, it would be the last thing done in the body (a so-called tail call). Now consider a non-void function definition of the form

```
290 def func(param_1, param_2, ...):
291     body # containing 'return expr' in tail positions only
```

A call of this function occurs as a (sub)expression, which is part of some statement, such as for example

```
292     print(func(expr_1, expr_2, ...) + 1)
```

In general, such a call takes the form

```
293     stmt(func(expr_1, expr_2, ...))
```

where `stmt` is a void function. Assuming, for simplicity, that the argument expressions have no side effects, it can be eliminated as follows.

1. Replace `stmt(func(expr_1, expr_2, ...))` by `body`.
2. Replace every occurrence of **return** `expr` in `body` by `stmt(expr)`. N.B. The expression may need to be parenthesized as `(expr)`.
3. Simultaneously replace all occurrences of parameters `param_i` by their corresponding argument expressions `(expr_i)`.
4. Systematically rename any local variables in `body` whose name clashes with a name occurring in the context of the call.

If a return statement would not occur in a tail position, then a ‘jump’ to the end of the body would also be needed. Note, however, that Python does not support `goto` statements (except on April’s Fool Day 2004).

Here is an example involving the famous recursive factorial function:

```
294 def fac(n: int) -> int:
295     """For n >= 0, return n factorial.
296     """
297     if n == 0:
298         return 1
299     else:
300         return n * fac(n - 1)
```

² Non-void functions are sometimes known as fruitful functions. In Python, they contain **return** `expr`.

Note that the two return statements occur in tail positions. Let's rewrite the statement `print(fac(2))`:

```

301     print(fac(2))
302
303 # inline call: n = 2
304
305     if 2 == 0:
296         print(1)
297     else:
298         print(2 * fac(2 - 1))
299
300 # delete dead if-branch: 2 != 0 ; simplify: 2 - 1 == 1
301
302     print(2 * fac(1))
303
304 # inline call: n = 1
305
306     if 1 == 0:
307         print(2 * 1)
308     else:
309         print(2 * 1 * fac(1 - 1))
310
311 # delete dead if-branch: 1 != 0 ; simplify: 1 - 1 == 0
312
313     print(2 * 1 * fac(0))
314
315 # inline call: n = 0
316
317     if 0 == 0:
318         print(2 * 1 * 1)
319     else:
320         print(0 * 2 * * 1 * fac(0 - 1))
321
322 # delete dead else-branch: 0 == 0 ; simplify: 2 * 1 == 2
323
324     print(2 * 1 * 1)

```

In general, `print(fac(n))` rewrites to

```

335     print(n * (n - 1) * ... * 2 * 1 * 1)

```

where there are $n + 1$ factors in the expression.

If the body of a non-void function consists of single return statement, then one can do equational reasoning directly on the level of expressions rather than statements. For instance, consider the following definition of function `fac`:

```
336 def fac(n: int) -> int:
337     return 1 if n == 0 else n * fac(n - 1)
```

Now, rewriting call `fac(2)` is less complicated:

```
338     fac(2)
339
340 # inline call: n = 2
341
342     1 if 2 == 0 else 2 * fac(2 - 1)
343
344 # delete dead if-branch: 2 != 0 ; simplify: 2 - 1 == 1
345
346     2 * fac(1)
347
348 # inline call: n = 1
349
350     2 * (1 if 1 == 0 else 2 * 1 * fac(1 - 1))
351
352 # delete dead if-branch: 1 != 0 ; simplify: 1 - 1 == 0
353
354     2 * 1 * fac(0)
355
356 # inline call: n = 0
357
358     2 * 1 * (1 if 0 == 0 else 0 * 2 * 1 * fac(0 - 1))
359
360 # delete dead else-branch: 0 == 0 ; simplify: 2 * 1 == 2
361
362     2 * 1 * 1
```

Appendix B. Better Python Solutions for Bit Strings

For completeness sake, let me note that the problem of printing all bit strings of a given length can be solved in a better way. Decompose the problem into:

1. Constructing all bit strings of a given length.
2. Printing them all.

In Python, one might be tempted to construct all those bit strings, by putting them in a list. But then they are all stored in memory before printing them (or doing whatever one wants, for instance, count them). For that reason, generator expressions were introduced in Python. They allow on-demand construction. Here is a Python solution based on Fig. 2 (middle):

```

363 from typing import Iterator
364
365 def generate_bit_strings(n: int) -> Iterator[str]:
366     """Yield all strings over "01" of length n,
367     in lexicographic order.
368
369     Assumption: n >= 0
370     """
371     if n == 0:
372         yield ""
373     else: # n > 0
374         yield from (b + u
375                     for b in "01"
376                     for u in generate_bit_strings(n - 1)
377                     )

```

It can be invoked to print the bit strings like this:

```

378 print(*generate_bit_strings(3), sep='\n')

```

And here is a solution based on Fig. 2 (right):

```

379 def generate_bit_strings(n: int) -> Iterator[str]:
380     if n == 0:
381         yield ""
382     else: # n > 0
383         yield from (u + b
384                     for u in generate_bit_strings(n - 1)
385                     for b in "01"
386                     )

```

Appendix C. Haskell Solutions for Bit Strings

It is illustrative to see the same definitions in a pure non-strict functional programming language like Haskell. Using the recursive decomposition in Fig. 2 (middle), combining two recursively grown trees of size one smaller:

```

387 bitStrings :: Int -> [String]
388 -- bitStrings n = list of strings over "01" of length n (n >= 0),
389 -- in lexicographic order
390 bitStrings 0 = [""]
391 bitStrings n = [b : u | b <- "01", u <- bitStrings (n - 1)]

```

and when growing one tree of size one smaller and splitting all its leaves (Fig. 2, right):

```

392 bitStrings 0 = [""]
393 bitStrings n = [u ++ [b] | u <- bitStrings (n - 1), b <- "01"]

```

The latter is not efficient, because appending at the end of a list is not efficient in Haskell. But this can be improved by introducing an accumulation parameter:

```

394 gbitStrings :: [String] -> Int -> [String]
395 -- gbitStrings s n = [t ++ u | t <- bitStrings n, u <- s]
396 -- hence, gbitStrings [""] = bitStrings
397 gbitStrings s 0 = s
398 gbitStrings s n = gbitStrings [b : t | b <- "01", t <- s] (n - 1)

```

Observe that this definition is more efficient, because it now prepends to a list. Moreover, it is tail recursive, and thus the recursion can be compiled into a loop. Also note that in Haskell, lists are lazy, that is, they are only constructed in so far as needed (like the generator expressions in Python used in Appendix B).

Trends in Teaching Programming in Schools in Hungary

Márton VISNOVITZ, Győző HORVÁTH

Eötvös Loránd University, Budapest

e-mail: visnovitz.marton@inf.elte.hu, horvath.gyozo@inf.elte.hu

Abstract. Programming education in Hungary has undergone significant changes with the new National Core Curriculum released in 2020. It introduced a new, revised Digital Culture curriculum in public schools as a successor to the earlier informatics subject. The new curriculum contains several new themes and topics, with a bigger emphasis on programming and algorithms. However, little is known about the effect of these changes on the teaching practices and tools used to teach programming. In this paper, we present the results of a survey conducted with schoolteachers, and data provided by the Educational Authority of Hungary. We identify the most common programming languages, environments and pedagogical methods used by teachers, to give a general overview of the trends in teaching programming in Hungarian public schools.

Keywords: programming, programming languages, teaching strategies, CS curriculum, Hungary.

1. Introduction

In Hungary the content of education in public schools is determined by the National Core Curriculum¹ (thereinafter NCC), a document issued by the government. The first NCC was issued in 1995, new versions were released in 2003, 2007, 2012, and most recently in the year 2020. While the NCC from 2012 is still in effect for students who started their current level of education before 2020, the latest 2020 NCC is already being implemented as well.

Informatics as a standalone subject was present in the NCC since its first 1995 edition. Even though the contents of the informatics subject went through a lot of changes through the years, the contents of the 2012 NCC was considered to be outdated soon after its release (Zsakó and Horváth, 2017). Textbooks designed for the 2012 NCC (Farkas, 2011; Rozgonyi-Borus and Kokas, 2018) had very little on algorithms and programming in general. In addition to some introductory programming with the Logo

¹ In Hungarian: Nemzeti Alaptanterv, or NAT for short.

programming language, these textbooks only included code examples in the Basic and Pascal programming languages.

In 2020 the new NCC introduced a lot of changes to the informatics education landscape in Hungary. In addition to new themes, topics, and concepts, it contained changes to the high school final exam as well. The school subject itself has been renamed from informatics to digital culture. With the new NCC also came new textbooks for the digital culture subject. While in the past teachers were allowed to choose from several state-approved textbooks and exercise books, the new 2020 NCC has a single textbook for each subject for the given year and school type. With this change teachers no longer have the liberty to choose their preferred book for their classes. The new textbooks cover a more diverse set of programming topics than their predecessors. In addition to classic algorithmic programming, they showcase modern tools and environments like Scratch or Micro:bit boards. For advanced years the programming language of choice in these textbooks is Python.

In Hungary, the final exams (or graduation exams) are the final test for high school students before they go to university. Their score on these exams forms the basis of the score for applying to college. In addition to the contents of education, the NCC and related documents specify the contents of the final exams for each school subject. When it comes to the programming task of the final exam, both the current and the previous regulations allow the students to choose from several programming languages and environments. This provides a wide variety of options for teachers when it comes to choosing the programming language to use for teaching programming for their students.

Considering the changes in the National Core Curriculum and the overall changing landscape of programming education in Hungary, we decided to conduct research on the methods and tools that are being used in the country, as well as the changes and current trends in teaching programming.

2. Research Method

To create a map of teaching methods and technologies (i.e., programming languages, programming environments and other educational tools) that are currently being used in Hungary, a nationwide online survey was conducted with teachers who teach informatics, digital culture, or some other related subject in schools. Questions in the survey focused on the programming languages and environments the teacher uses, as well as the type of tasks they choose for their classes. We also asked what tools they used earlier but decided to abandon, to have a better understanding of the changes in the choice of tools.

We wanted to get differentiated information about the methods used for students of various age groups, thus our survey consisted of similar questions for years 1–4, years 5–6, years 7–8, years 9–10, and years 11–13. Hungarian school is twelve or thirteen years (some schools have an extra year for intensive language courses), usually divided into three stages. Years 1–4 is elementary school, years 5–8 is primary school, and years 9–12 is secondary school or high school.

To have a better understanding of the demographic distribution of the respondents, we also asked for anonymous data about the teachers themselves. These questions were about the type and location of the school where they teach, their level of education in teaching/pedagogy, and information about their years of experience teaching informatics-related subjects.

To complement the data collected with the survey, we also contacted the Educational Authority² of Hungary to request information on the programming languages and programming environments used by students on the final exams in informatics. The data we received about the final exams from 2013 to 2020 also provides some insight on the tools and methods that are being used to teach programming, even though we cannot observe the changes induced by the NCC of 2020 on this dataset.

3. Survey Results

The online survey was filled out by a total of 169 teachers. According to the Educational Authority and the National Institute of Vocational and Adult Education³ at the time of the survey there were a total of 8038 teachers in Hungary who were teaching informatics or some other informatics-related subject. Of this total number, 4702 teachers work in general curriculum schools, and 3338 teachers work in vocational schools. Data from the survey responses was aggregated and analyzed. Where applicable, data entered in the freeform fields was merged with the responses for the predefined answers. Other information in freeform answers was processed manually to gain more insight on the trends and the reasoning for changes.

Demographic Data

As seen on Fig. 1a, respondents are evenly distributed between Budapest (capital of Hungary, 34%), county capitals (34%) and other cities (30%). With only four respondents, smaller towns and villages are not well represented (2%). As the 2012 NCC had no informatics subjects for the first four years of school, it is possible that many of the elementary schools in these smaller settlements don't employ informatics teachers at all.

Most teachers who filled out the survey (84%) have a university degree as opposed to those who possess a college degree (12.4%). While in the past teacher training was available on the college level, since 2006 all teacher training programs grant a university degree. This is well represented by that data about the respondents' academic training as seen in Fig. 1b.

² In Hungarian: Oktatási Hivatal.

³ In Hungarian: Nemzeti Szakképzési és Felnőttképzési Hivatal.

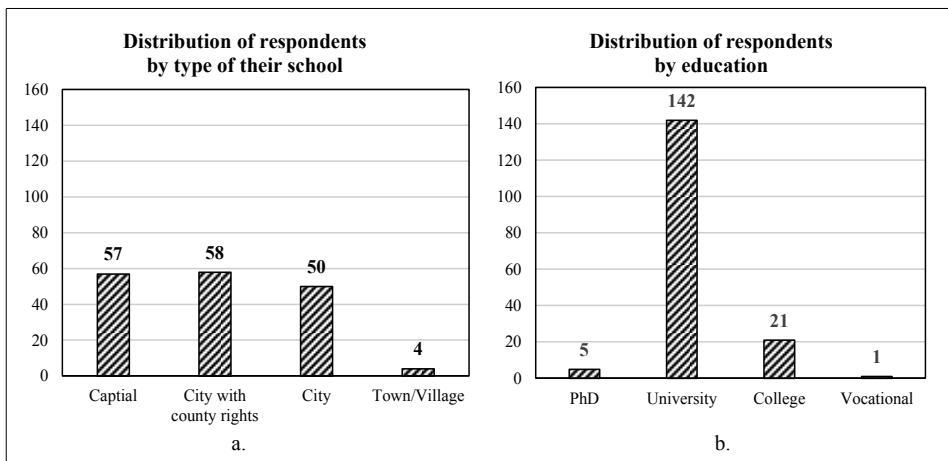


Fig. 1. Distribution of respondents based on their type of school (a) and their education (b).

Programming Languages and Environments

The biggest part of our online survey contained questions about programming languages and environments. We asked teachers to indicate what tools they use to teach programming to students of certain ages. As our default options we selected a mix of visual and code-based programming environments, but they also had the opportunity to add new options to the list.

As seen in Fig. 2 the most popular programming environment between years 1 through 8 is Scratch (with adoption rates of 62%, 81% and 49%), but it is still popular

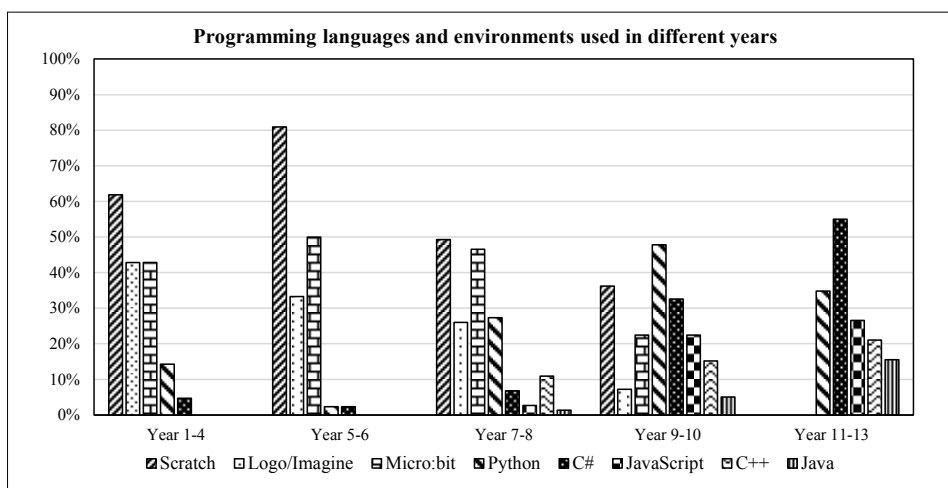


Fig. 2. Programming languages and environments used for teaching programming for different age groups.

in years 9 and 10 (36%). Another popular choice throughout the years are Micro:bit boards. They are most popular also from years 1 to 8 (43%, 50%, 47%), but still used by some in years 9 and 10 (22%). This correlates with parts of the content of new textbooks for the NCC of 2020 (Abonyi-Tóth *et al.*, 2020; Abonyi-Tóth *et al.*, 2022, 2023; Lénárd *et al.*, 2020) that among others use these two environments for introductory programming for younger students. While Logo and turtle graphics is not present in the new elementary school textbooks (Lénárd *et al.*, 2021, 2022), it is still popular in the early years (43%, 33%, 26%), but it loses significance by the later years. Textbooks for these ages focus on teaching algorithmic thinking by the programming of virtual and physical robots.

As classical algorithmic programming goes, the new textbooks for years 9 to 11 (Abonyi-Tóth *et al.*, 2020; Abonyi-Tóth *et al.*, 2020; Varga *et al.*, 2020) use Python almost exclusively. Data from the responses show that Python is popular among teachers in these years (48% and 35%), but some teachers also use it earlier as well (27% in years 7–8). Other popular languages in the high school years are C# (33% and 55%), and C++ (15% and 21%). Java, a programming language that is also available for the final exams has lower numbers (5% and 16%). An interesting outlier in the later years is JavaScript, a language that cannot be used on the final exams, but it is still the 3rd most popular code-based language based on responses (22% and 27%). It is also worth noting, that Scratch is also used extensively in years 9 and 10 (36%), even though textbooks for these years already drop block programming in favor of code-based programming.

In our survey we also asked teachers about the programming languages and environments that they used in the past but have abandoned for some reason. We wanted to find out what are the tools and methods that teachers decided not to use anymore, and what was their reasoning to do so. Many respondents (roughly an average of 45% between years 7–13) said that they abandoned Pascal as they see it outdated compared to more modern options. Another language/environment that a lot of teachers mentioned in this category is Logo. Roughly 50% of teachers said that they used Logo in the past between years 1 and 8 but they decided to drop it. Many of these respondents said that they stopped using Logo in favour of Scratch.

Strategies for Teaching Programming

In addition to information about programming languages and environments, we also wanted to learn more about the types of tasks teachers use to teach programming to students of certain ages. We based our options on the strategies identified by Bernát and Zsakó (Bernát and Zsakó, 2017). These strategies include teaching programming and algorithmic thinking through turtle graphics, robotics, everyday algorithms, the creation of animations and graphical games, fundamental algorithms (mathematics-based) and application development (desktop or mobile).

As seen on Fig. 3, turtle graphics and robotics are popular methods in years 1 through 8. Even though Logo is no longer directly a part of the digital culture curriculum, it still

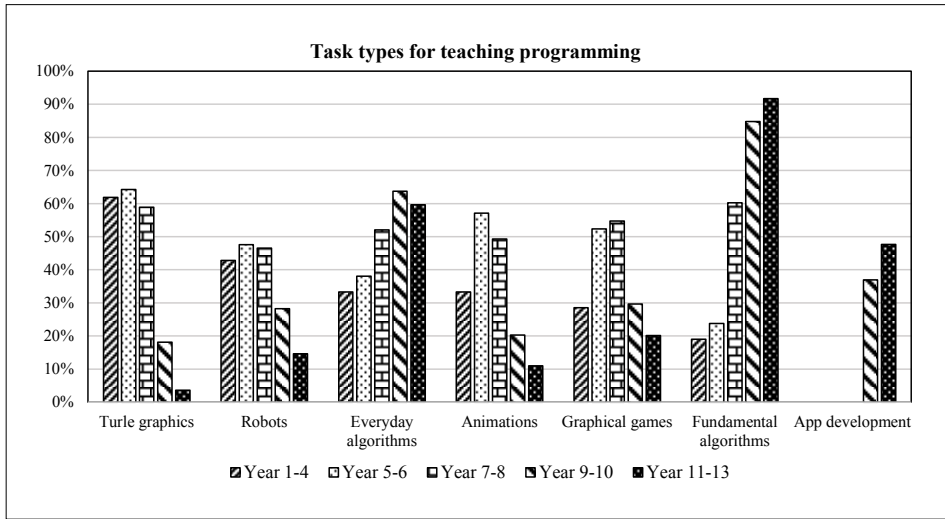


Fig. 3. Task types used for teaching programming in different years.

has around 60% adoption rate in these years (62%, 64%, and 59%). This could mean that teachers found new tools to teach using turtle graphics, as several respondents said that they abandoned Logo in favor of Scratch. The high numbers for the usage of robots (43%, 48%, and 47%) in these years aligns well with the new NCC of 2020 in which robotics gets a lot of focus in the early years.

The usage of everyday algorithms to teach programming is significant throughout all years. Interestingly it seems to be more popular in high school than in earlier years (64% and 60% in high schools compared to the 33%, 38% and 52% in elementary and primary school). The development of complex programs with graphical user interfaces is also used by a significant number of teachers in high school (37% and 48%⁴), but by far the most used method in this age group is the use of fundamental algorithms or programming theorems (Gregorics *et al.*, 2019; Szlávi *et al.*, 2019). Its use gets significant in years 7 and 8 (60%) and is used by almost all teachers in later years (85% and 92%). This is expected as the most popular programming competitions in Hungary as well as the programming tasks of the final exam focus almost exclusively on the usage of fundamental algorithms.

The creation of animations and graphical games teaching strategies for programming is most popular between years 5 and 8 with approximately half of respondents saying they use these approaches (57% and 49% for animation and 52% and 55% for games). Scratch, a popular tool for this age group can be used for both, so it can be a good choice for teachers who want to use these methods in their classrooms. The textbooks of the 2020 NCC also explore animations and games to a degree, but focus more on Micro:bit boards for this age group.

⁴ Application development was not listed as an option for years 1–8.

4. Data on Final Exams

In Hungary final exams are organized biannually with one exam in the summer (May–June) and another in the autumn (October–November). As the normal time to take the final exams for students who finish high school is the summer, the number of students who participate in the exams of the summer period is significantly higher than those who take the autumn exams (ca. 11 times as much for years between 2013 and 2020).

Before the final exams in informatics, participating students must fill out a preliminary form to indicate the programming environment they want to use on the exam. The Educational Authority of Hungary has provided us with anonymous data about the responses from these forms for the exams between 2013 and 2020. The data we received contains information about both exam periods for each year, but as summer exams have a lot more participants, we opted to compare data for these exams to determine trends. Also, it is worth noting that the last year we have information about is 2020, so the changes triggered by the new 2020 NCC are not visible in the data.

On the form students only specify a programming environment, not the actual programming language. This means that we cannot get exact information about the programming languages used on the exam, but the environment is a good indicator of the language used. This means that for the analysis we grouped some programming languages together (e.g., C#, Visual Basic and Visual C++) as it is not possible to determine which one was used based on the data available. On the other hand, some environments in the list clearly indicate the programming language by listing the compilers available. We also have no data about the programming languages chosen by students who decided to take the exams on a Linux environment rather than Windows, as this is listed as a single environment on the form. The number of these students is very low (~0.6%) so omitting this data does not change the overall trends significantly.

As seen on Fig. 4, with consistently over 40% of students choosing it, the most popular programming languages on the final exams is the Visual Basic/C#/C++ group. While

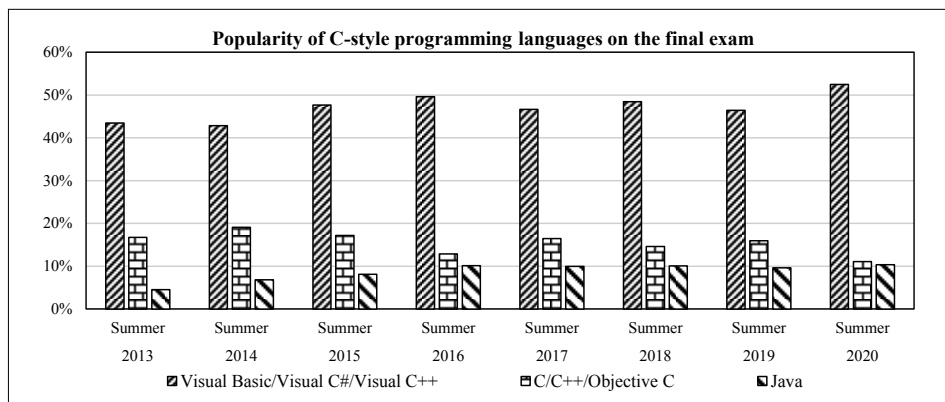


Fig. 4. Programming environments chosen by students on the summer final exams (2013–2020).

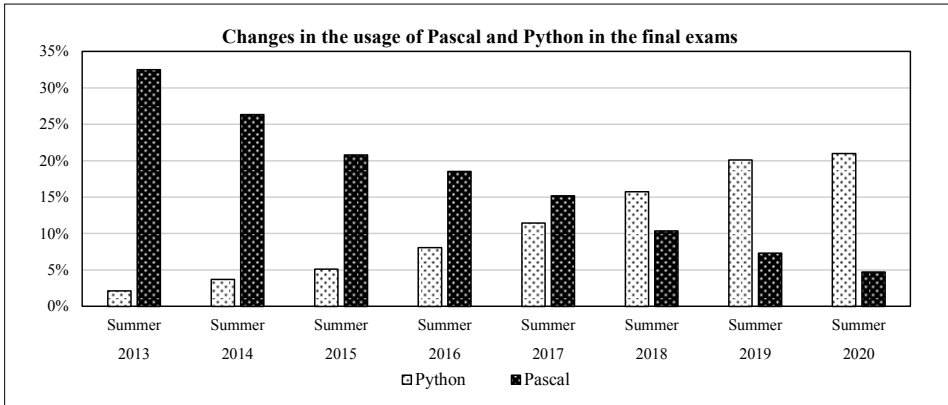


Fig. 5. The change in the popularity of Python and Pascal on the final exams (2013–2020).

based on the form responses we cannot tell which of the three indicated languages was used by the student, based on the results of the teacher survey it is logical to assume that most of the students in this category chose C# to solve the tasks of the final exam. This choice of environment shows a slight increasing trend in popularity with an average of about 1% per year increase in the investigated period.

C, C++, and Objective C are also in the same category based on the available options on the final exam form. These languages were selected by an average of 16% of the students with a very slight ($\sim 0.7\%$ /year) decreasing trend between 2013 and 2020. With a similarly slight average change ($\sim 0.7\%$ increase/year) Java is not particularly popular with an average of 9% of the students choosing it for the exams.

As seen on Fig. 5, the most significant changes in popularity can be seen in the usage of Pascal and Python in the given years. While popular in the past, the selection rate of Pascal dropped from 33% to only 5% in seven years. At the same time the number of students choosing Python has increased from 2% to 24% making it the second most popular choice. This trend is also indicated by the number of teachers who said in the survey that they use Python, and the high number of educators who said that they no longer use Pascal to teach programming.

5. Conclusions

With the release of the new National Core Curriculum in 2020 a lot of changes were introduced to the landscape of programming education of Hungary. With a bigger focus on algorithms and programming in the new curriculum we conducted an investigation on what programming languages, environments and task types are used by teachers in the country to teach programming, as well as the trends in the usage of programming languages in schools and the final exams.

Based on a nationwide survey conducted with teachers, we found that Scratch is a very popular tool in the country to teach programming, especially in elementary and pri-

mary schools. While Scratch is also used in high schools, the most popular languages for students of this age group are C# and Python. These languages are popular choices in the final exams as well. JavaScript is also popular in high schools, even though its use is not allowed on the final exams in informatics or digital culture. Other popular educational tools include Logo and Micro:bit boards. They are widely used to teach programming between years 1 and 8.

As for types of tasks, turtle graphics, robotics, and the creation of animations and visual games are the most popular methods to teach programming in the early years. In high schools tasks based on fundamental algorithms are dominant, with application development also being a relevant method. While used in every age group, the usage of everyday algorithms to deepen students' understanding of algorithms seems to be more popular in later years.

On the final exams between 2013 and 2020 the most popular programming environment of choice for students was Visual Studio. This indicates the popularity of C# for solving the programming tasks of the exam. The usage of Pascal is in a rapid decline in the final exam, while Python is becoming more and more popular.

While the data we collected through our survey and the data received from the authorities helped us to have a better understanding of how we teach programming in the schools of Hungary, due to the recent modifications in the framework of informatics education, it is important to repeat this research in the upcoming years to see what changes were caused in the trends in teaching programming.

Reference

- Abonyi-Tóth, A., Farkas, C., Fodor, Z., Jeneiné Horváth, K., Reményi, Z., Siegler, G., Varga, P. (2020). *Digitális Kultúra 11*. Oktatási Hivatal.
- Abonyi-Tóth, A., Farkas, C., Jeneiné Horváth, K., Reményi, Z., Tóth, T., Varga, P. (2020). *Digitális Kultúra 10*. Oktatási Hivatal.
- Abonyi-Tóth, A., Farkas, C., Turzó-Sovák, N., Varga, P. (2020). *Digitális Kultúra 6*. Oktatási Hivatal.
- Abonyi-Tóth, A., Farkas, C., Varga, P. (2022). *Digitális Kultúra 7*. Oktatási Hivatal.
- Abonyi-Tóth, A., Farkas, C., Varga, P. (2023). *Digitális Kultúra 8*. Oktatási Hivatal.
- Bernát, P., Zsakó, L. (2017). Methods of Teaching Programming – Strategy. *XXXth DIDMATTECH 2017*, 40–51.
- Farkas, C. (2011). *Informatikai ismeretek a 7. évfolyam részére* (B. Danitz (ed.)). Jedlik Oktatási Stúdió.
- Gregorics, T., Kovácsné Pusztai, K., Fekete, I., Veszprémi, A. (2019). Programming theorems and their applications. *Teaching Mathematics and Computer Science*, 213–241. <https://doi.org/10.5485/TMCS.2019.0466>
- Lénárd, A., Abonyi-Tóth, A., Turzó-Sovák, N., Varga, P. (2020). *Digitális Kultúra 5*. Oktatási Hivatal.
- Lénárd, A., Sarbó, G., Tarné, É. (2021). *Digitális Kultúra 3*. Oktatási Hivatal.
- Lénárd, A., Turzó-Sovák, N., Tarné, É., Sarbó, G. (2022). *Digitális Kultúra 4*. Oktatási Hivatal.
- Rozgonyi-Borus, F., Kokas, K. (2018). *Informatika 8*. (K. Tóth (ed.)). Mozaik Kiadó.
- Szlávi, P., Zsakó, L., Törley, G. (2019). Programming theorems have the same origin. *Central-European Journal of New Technologies in Research, Education and Practice*, 1(1), 1–12. <https://doi.org/10.36427/cejntrep.1.1.380>
- Varga, P., Jeneiné Horváth, K., Reményi, Z., Farkas, C., Takács, I., Siegler, G., Abonyi-Tóth, A. (2020). *Digitális Kultúra 9*. Oktatási Hivatal.
- Zsakó, L., Horváth, G. (2017). Quo Vadis, Informatics Education? – Towards a more up-to-date informatics education. *Acta Didactica Napocensia*, 10(3), 45–52.



M. Visnovitz is a PhD student at Eötvös Loránd University, Budapest at the Faculty of Informatics since 2018. He also obtained his degree here as a teacher of informatics and environmental sciences. Between 2017 and 2021 he worked as an assistant lecturer at the Department of Media and Educational Informatics, his primary courses were on web programming and teaching methodology. His research focuses on the usage of the web as a programming platform to teach programming in schools.



G. Horváth is an associate professor at Eötvös Loránd University, the head of the Department of Media and Educational Informatics since 2022. He obtained his PhD in physics in 2009 from the Faculty of Science of Eötvös Loránd University as well. His main research topic is methods of introductory programming, programming methodology and web technologies in programming education. He is also actively participating in preparing students for international programming competitions, such as the IOI.

REPORTS

Secondary School Programming Olympiads in Gomel Region

Michael DOLINSKY,

*Faculty of Mathematics and Technologies of Programming, F. Skorina Gomel State University
Sovetskaya str., 104, Gomel. 246019. Republic of Belarus
e-mail: dolinsky@gsu.by*

Abstract. This article describes the content of programming competitions for students in grades 5–8 of the Gomel region. A general idea of the content of the tasks and examples of tasks by topic are given. The methodology for teaching and preparing schoolchildren for such Olympiads is also briefly described. A serious technical basis is the instrumental system of distance learning developed under the supervision of the author (<http://dl.gsu.by>).

Keywords: programming olympiads, secondary school, instrumental system of distance learning.

1. Introduction

Nowadays programming starts in elementary school (Dagienė *et al.*, 2019). It then continues into secondary school in a variety of ways: it could be machine-less learning (Pluhar, 2021; van der Vegt, 2016), game learning (Combefis *et al.*, 2016), using Scratch (Fagerlund *et al.*, 2020), using of specialized software development environments (Kabátová *et al.*, 2016; Tsvetkova *et al.*, 2021; Alemany *et al.*, 2016), robot programming (Kanemune *et al.*, 2017; Panskyi *et al.*, 2021).

Since September 1996, on the basis of secondary school 27 in Gomel, and in September 1999, additionally and on the basis of the distance learning site DL.GSU.BY (hereinafter referred to as DL), work is being carried out on the optional study of computer science and programming for schoolchildren of different ages (Dolinsky, 2016). The key feature of this training is the early start of education – actually from the 1st

grade, and in some cases from kindergarten (Dolinsky, 2018). For such students, special programming olympiads are held in order to increase motivation for classes, as well as for the early acquisition of competitive experience. Problems for Olympiad in programming for 1–4 grades students of primary school are described at (Dolinsky, 2022). Verification of solutions is carried out automatically on the DL.GSU.BY website (Dolinsky, 2017). This article offers materials for programming Olympiads for students in grades 5–8 of secondary school and a brief description of teaching programming and preparing for such Olympiads.

Training includes a consistent study of the necessary information and fixing them by solving the proposed problems from the systematically collected problems of Olympiads of past years. Solutions are checked automatically on the DL.GSU.BY website. Note that all the Olympiad tasks of the current academic year are included in the systematic training immediately after the last Olympiad. We also note how ideas for new problems appear. On the one hand, we focus on the IOI-curriculum, and on the other hand, every year we solve the problems of the USACO, COCI, St. Petersburg individual and team Olympiads. It is they (or their subtasks) that serve as a source for the tasks of our Olympiads in subsequent years.

2. Content of the Olympiads

Tasks for grades 5–8 include 10 tasks in ascending order of complexity (each student is invited to solve all these tasks) on the following topics:

1. Introduction to programming.
2. One-dimensional array.
3. Two-dimensional array.
4. Geometry.
5. Strings.
6. Sorting.
7. Text task.
8. Elements of number theory.
9. Greedy.
10. Queue.

Topic 1: Introduction to Programming

The topic **Introduction to Programming** includes tasks in which you need to enter the initial data and output the answer, that is, to solve them, you do not need to know anything other than input and output operators. At the same time, these tasks are used for Propaedeutics of knowledge that may be needed in the future, in mathematics, programming language, Olympiad programming.

At the moment, the topic “Introduction to Programming” contains tasks on the following subtopics: formatted output; algebraic formulas; numeric operations (AND, OR,

XOR, DIV, MOD, SHL,SHR); built-in functions and procedures (ABS, SQR, ODD, ROUND, TRUNC, ORD, CHR, UPCASE, STR, VAL, LENGTH, COPY, DELETE, INSERT, POS); number systems.

An example of a task on the topic “Introduction to Programming”:

Problem “Partial and Remainder”

Sample Input	Sample Input:
7 2	6 4
Sample output:	Sample output:
7 div 2 = 3 7 mod 2 = 1	6 div 4 = 1 6 mod 4 = 2

Topic 2: One-Dimensional Array

The topic **One-Dimensional Array** includes tasks to solve that require knowledge of the one-dimensional array declaration, IF condition statements, FOR and WHILE loops. At the same time, it contains tasks both for standard algorithms for processing one-dimensional arrays and tasks for Propaedeutics of useful knowledge. Currently, the topic “One-Dimensional Array” contains tasks on the following subtopics: sum; count; maximum/minimum; maximum/minimum number; cycle range; even/odd positions; long arithmetic (addition, subtraction, multiplication of a number by a digit, signs of divisibility), prefix/suffix sums, maximums, minimums; sorting by counting; cycle parameters; cyclic account; deque; search; running in a row; finding of all different numbers.

An example of a task on the topic “One-Dimensional Array”:

Problem “Divisibility by 3”

Find out if a positive integer is divisible by 3. The number can have up to 30 digits and is given by an array A of N digits. Note: A number is divisible by 3 when the sum of its digits is divisible by 3. For example, the number 159 is divisible by 3 because the sum of its digits $1 + 5 + 9 = 15$ is divisible by 3. Print the remainder of the sum of its digits divided by 3 and Yes/No (divided or not given number by 3)

Input format:	Sample Input
N A[1] A[2] ... A[N] (numbers are entered with a space)	3 1 5 9
Output Format:	Sample output:
r – remainder after dividing the sum of digits by 3 Yes/No	0 Yes

Topic 3: Two-dimensional array

The topic **Two-dimensional array** includes tasks for solving which require additional knowledge of the declaration of a two-dimensional array and the use of nested loops. Currently, the topic “Two-Dimensional Array” contains tasks on the following sub-topics: sub-array – sum; sub-array – count; line count; array generation; array modification; counting along the perimeter of the array; prefix sums, maximums, minimums; strings comparison in array.

An example of a task on the topic “Two-dimensional array”:

Problem “Reset the maximum in rows”

A two-dimensional array of $N \times N$ elements is given. Zero out the first maximum element in each row.

Input Format:	Sample Input
N ($N \leq 10$) $a[1,1]$ $a[1,2]$... $a[1,N]$ $a[2,1]$ $a[2,2]$... $a[2,N]$... $a[N,1]$ $a[N,2]$... $a[N,N]$	5 3 4 1 3 8 2 5 6 6 5 1 3 6 1 4 3 5 1 7 2 1 2 3 2 1
Output Format:	Sample output:
$b[1,1]$ $b[1,2]$... $b[1,N]$ $b[2,1]$ $b[2,2]$... $b[2,N]$... $b[N,1]$ $b[N,2]$... $b[N,N]$	3 4 1 3 0 2 5 0 6 5 1 3 0 1 4 3 5 1 0 2 1 2 0 2 1

Topic 4: Geometry

The topic **Geometry** includes tasks that require the ability to solve such problems as find the distance between: two points, a point and a set of points, all points of the set, and then apply the previously studied algorithms on one-dimensional and two-dimensional arrays. In addition, this topic includes tasks on such basic concepts of geometry as perimeter and area. Currently, the topic “Geometry” contains tasks on the following subtopics: rectangle, Manhattan distance, distances from one point to set of points, the distance between the same points of two sets; neighboring distances; distances between all pairs of points; distances between all pairs of points of two sets; polygon area.

An example of a task on the topic “Geometry”:

Problem “Area under the segment”

The segment is given by the coordinates of its ends (x_1 , y_1) and (x_2 , y_2).

Determine the area of the figure bounded by this segment, the vertical lines from its ends and the X axis.

Notes.

1. The educated figure is a trapezoid.
2. The area of a trapezoid is half the sum of the bases times the height.

Input Format:	Sample Input
x1 y1 x2 y2	2 2 1 1
Output Format:	Sample output:
S output the answer with one decimal place	1.5

Topic 5: Strings

The topic **Strings** includes tasks for the solution of which you need to know the data types character, string, array of strings and be able to invent and debug your own algorithms. In fact, this topic is key to determining the potential abilities of the student. Currently, the topic “Strings” contains tasks on the following subtopics: cyclic shift to the right; if; string reversal; count on string; maximum per string; using of ORD; search in a string; bracket strings; lengths of array strings; count in array of strings; strings generation; strings array generation; converting a sentence into an array of words; formation of arrays of strings; subarray of characters; strings array editor; analysis of all cyclic shifts of the string.

An example of a task on the topic “Strings”:

Problem “Maximum match”

N, A, B are given. A and B are strings of the same length, N is the length of these strings. Among all cyclic shifts to the right of string A, choose the one that matches the positions of the maximum number of characters with string B. Output this maximum number.

Input Format:	Sample Input	Explanations
L S1 s2	4 AGTC CTGA	
Output Format:	Sample output:	
Max	2	CAGT CTGA

Topic 6: Sorting

The topic **Sorting** includes tasks on the ability to apply the sorting algorithm by exchange, bubble, counting and includes tasks on the following subtopics: sorting only; sorting and output element with fixed number; fixed numbers; post-condition after sorting; sorting and output elements with variable numbers; sorting and output elements from variable range of numbers; sorting with numbers; sorting by counting; compression of coordinates; all different in ascending order.

An example of a task on the topic “Sorting”:

Problem “Compression of coordinates on a straight line”

Compress coordinates of points on a straight line. All coordinates are different integers from -10^9 to 10^9 . The essence of coordinate compression is as follows: – sort all points by coordinate. instead of the coordinates of their numbers.

Input Format:	Sample Input:
K – number of numbers ($k \leq 10$) a[1] a[2] ... a[K] – numbers separated by spaces – coordinates before compression	4 1000 -2000 -1000 2000
Output Format:	Sample output:
n[1] n[2] ... n[K] – coordinates after compression	3 1 2 4

Topic 7: Text problem

The topic **Text problem** contains problems with the original texts of the conditions of problems from the Belarusian Republican or Regional Olympiad (usually 1–2 pages), in which the task is changed so that its completion does not require knowledge more than that used in solving problems from topics 1–6. The main difficulty for the participant is to single out the algorithmic statement of the problem from the textual condition. At the moment, topic 7 includes subtasks on the following algorithmic subtopics: one-dimensional array: sum, sum + condition, count, maximum, maximum number, minimum, minimum number, conditional sum, conditional minimum, selection of elements, counting sort, divisors, loops; two-dimensional array: number of maximum in a column, row sum, maximum in an array, adjacency matrix; array of strings: count by array; line count.

An example of a task on the topic “Text task”:

Task “Martian games (simplified)”

One of the most important sporting events on Mars is coming soon – the Martian Games! Athletes from all regions of Mars take part in them. Sergei Petrovich, coach of the Olymp City team, is also preparing his athletes for the Games. He trains n athletes,

k of which will go to the Games. The coach faced a difficult question: it is necessary to decide who will get into the national team. Competitions are held on Mars in two stages: steeplechase and swimming, with each athlete going through both stages. The coach calculated that the i -th athlete could run in a_i minutes and swim in b_i minutes.

There is one feature at the Mars Games: the personal results of the athletes are practically not taken into account, and the penalty time of the team plays the main role. In competitions, the penalty time of the team is calculated according to the following formula: the product of the total running time by the total time for which all team members swam the distance. The smaller the penalty time of the team, the higher the place.

For example, let there be three athletes who ran in 3, 8, and 5 minutes, respectively, and swam in 4, 9, and 1 minute. Then the team penalty time is $s = (3 + 8 + 5) * (4 + 9 + 1) = 16 * 14 = 224$.

Sergey Petrovich wants to choose k athletes in such a way as to minimize the team's penalty time.

Your task is much simpler: print the number of the first athlete with the minimum total running and swimming time.

Input format

The first line of the input contains two integers n and k ($1 \leq k \leq n \leq 2000$) – the number of Sergey Petrovich's athletes and the required team size for the Mars Games.

Each of the next n lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq 10^6$) – the number of minutes during which the i -th athlete will be able to run and swim the distance.

Output format

Print the number of the first athlete with the minimum total running and swimming time.

Example

Input	Output
10 8 12 1 13 4 1 33 10 10 3 6 1 19 3 12 10 10 7 7 33 2	5

Topic 8: Elements of number theory

The topic **Elements of number theory** includes tasks that require a preliminary study of the relevant theory. Currently, the topic “Elements of Number Theory” includes sub-

tasks on the following topics: For loop; nested For loops; while loop; For + While; dividers; simplicity test by definition; sieve of Eratosthenes; number systems; bit processing; submasks.

An example of a problem on the topic “Elements of number theory”:

Problem “All submasks”

You are given a positive integer i ($i < 1000$). It is required to display in descending order all submasks of the binary representation of this number, starting from the number itself.

Input Format:	Sample Input:
I	19
Output Format:	Sample Output:
Ib	19 10011
I1 b1	18 10010
i2 b2	17 10001
...	16 10000
Here	3 00011
Ik – decimal number in three positions	2 00010
bk is its bit representation	1 00001
(in the same number of bits as the original number)	

Topic 9: Greedy algorithm

The topic **Greedy algorithm** includes tasks for the solution of which it is required to pre-sort the input data. Currently, this topic includes tasks on the following subtopics: quadratic sorting; sorting with numbers; sorting by counting; quick sort; quick sort + while; quick sorting with numbers; comparison function; sorting array from numbers 1, 2, 3; correct bracket sequence; maximum depth of bracket expression; two arrays; selection of applications; deadline and price; minimum coverage; enumeration + greedy; stack + greedy; ad hoc.

An example of a problem on the topic “Greedy Algorithm”:

Problem “Applications”

Given N applications for conducting classes in a certain audience. In each application, the beginning and end of the lesson are indicated (s_i and f_i , respectively, for the i -th application). In the case of intersection of requests, only one of them can be satisfied. Requests with numbers i and j are joint (do not intersect) if the intervals $[s_i, f_i]$ and $[s_j, f_j]$ do not intersect (that is, $f_i \leq s_j$ or $f_j \leq s_i$). The task is to collect the maximum number of not intersected applications.

Input format:

N

$S[1] F[1]$

S[2] F[2]

...

S[N] F[N]

Where:

N – Number of orders.

S[i] F[i] – description of the i-th order.

Restrictions:

$1 \leq N \leq 200,000$

$0 \leq S[i] < F[i] \leq 100,000,000$

All numbers are integers.

Output Format:

Ans – the answer to the problem – the maximum number of not intersected applications

Example:

Sample Input:	Sample output:
5 1 13 6 8 24 4 5 7 10	3

Topic 10: Queue

The topic **Queue** includes tasks for the solution of which it is necessary to know the theory on this topic. At the moment, this topic contains tasks on the following subtopics: horse; labyrinth; three-dimensional labyrinth; pieces; three-dimensional pieces; a horse with a dynamic list of moves; numerical sequences; research with help of queue; 01-BFS; queue with bit processing.

An example of a task on the topic “Queue”

Problem “Introvert”

Ruslan is an introvert. He doesn't like to socialize, but prefers to be alone. He lives in a cubidom with many rooms: K floors, K*K rooms on each floor. He constantly has to look for the most remote room from all the neighbors and guests. You are given a map of this cubidom, count how many moves Ruslan needs to make to reach the nearest guest or neighbor.

Input format:

The first line contains the number K ($1 \leq K \leq 100$). Next is the description of K floors separated by an empty line. Each floor is a K*K matrix. Symbol '*' = empty cell 'G' = guest 'S' = neighbor 'R' = Ruslan. Unambiguity and correctness of tests is guaranteed.

Output Format:

Print the minimum distance from Ruslan to the nearest guest or neighbor. You can move

from any room in 6 directions (left, right, forward, backward, up a floor, down a floor) if you don't go outside the house.

Example:

Sample Input:	Sample Output:
3 G** *** *** *** *** *** *** *** **R	6

Systematic and purposeful preparation of regional Olympiads is an important means of developing the Olympiad movement in the region. Regional Olympiads are held in the Gomel region five times in the academic year: in October–November, school and city grades 1–11, and in March–April, school, city and regional (zonal) for students in grades 1–9. When conducting these Olympiads, Internet technologies and the DL.GSU.BY website are used, which allows not only schoolchildren from the Gomel region, but also everyone to participate in all the Olympiads. And, it should be noted, there are dozens of such people from all regions of Belarus and Minsk.

3. Training and Motivation System

It is important to note that, despite the focus on programming, training is essentially developing in nature and therefore it is very useful both for those who later choose information technology as their professional field, and for everyone who will be engaged in at least some time. Practice also shows that training is built in a rather interesting form. All classes are conducted only on a voluntary basis during extracurricular time. Another equally important aspect is a differentiated teaching. The use of Internet technologies makes it possible to provide individual training along a personal educational trajectory. If the student is unable to solve the problem, he is consulted by other students or the teacher. Face-to-face classes are held on Wednesdays and Sundays on the basis of the computer science cabinet of secondary school 27 in Gomel. Additionally, all students can work from home, skipping tasks that they themselves cannot solve, in order to subsequently receive help on this task in a face-to-face lesson.

In addition, weekly on weekends from Friday 8.00 to Sunday 20.00, one of the regional Olympiads that took place earlier in 2010–2022 opens for solving, solving which (at a convenient time for himself) each student can check how well he knows the mate-

rial he has covered, and also what topics are still to be studied. The teacher receives similar information about each of his students.

We also note the presence of seasonal Cups (who solves more problems in learning for a certain period – autumn, winter, spring, summer from all students in grades 5–8). The awards ceremony for the top three students takes place on the first Sunday of the next season. Additionally, on the first Sunday of September, the “Person of the Year” is awarded – a student who solves the most tasks in learning for the entire academic year (autumn, winter, spring, summer). Prizes and diplomas are sent to non-resident winners by mail. These awards are provided by OpenMyGame (<http://OpenMyGame.com>), founded by graduates of our classes.

In addition, every season of the year, the student who has made the most progress during that period of time is awarded. This award is held by the leader of the circles for programming based on the use of DL in St. Petersburg (http://vk.com/spb_dl).

4. Conclusion

This article presents the materials of programming olympiads for students in grades 5–8 and briefly presents the methodology for teaching and preparing these students for such Olympiads. In April 2023, 3 secondary school students Gennady Martsinkevich (grade 6), Kirill Kardash (grade 7), Mikhail Brel (grade 8) became diploma winners of the Belarusian Republican Olympiad. And Mikhail Brel also got to the selection for IOI 2023. This became possible because they are ahead of the curve and participated in 9–11 grade Olympiads, which additionally contain the following topics: recursion; recurrent relations and dynamic programming; graphs; complex data structures; complex dynamic programming. In the future, we plan to include these topics in the Olympiads for grades 5–8.

References

- Aleman, F.J., Vilahur, V.J. (2016). eSeeCode: Creating a computer language from teaching experiences. *Olympiads in Informatics*, 10, 3–18.
- Combéfis, S., Beresnevičius, G., Dagienė V. (2016). Learning programming through games and contests: Overview, characterization and discussion. *Olympiads in Informatics*, 10, 39–60.
- Dolinsky, M. (2013). An approach to teach introductory-level computer programming. *Olympiads in Informatics*, 7, 14–22.
- Dolinsky, M. (2014). Technology for the development of thinking of preschool children and primary school children. *Olympiads in Informatics*, 8, 63–68.
- Dolinsky, M. (2016). Gomel training school for Olympiads in Informatics. *Olympiads in Informatics*, 10, 237–247.
- Dolinsky, M. (2017). A new generation distance learning system for programming and Olympiads in Informatics. *Olympiads in Informatics*, 11, 29–39.
- Dolinsky, M., Dolinskaya, M. (2018). How to start teaching programming at Primary School. *Olympiads in Informatics*, 12, 13–24.
- Dolinsky, M., Dolinskaya, M. (2019). Training in writing the simplest programs from early ages, *Olympiads in Informatics*, 13, 21–30.

- Dolinsky, M., Dolinskaya, M. (2020). The Technology of Differentiated Instruction in Text Programming in Elementary School Based on the Website dl.gsu.by, *Olympiads in Informatics*, 14, 37–46.
- Dolinsky, M. (2022). Primary School Programming Olympiads in Gomel Region (Belarus). *Olympiads in Informatics*, 16, 107–123.
- Fagerlund, J., Hakkinen, P., Vesisenano, M., Viiri, J. (2020). Assessing 4th Grade Students' Computational Thinking through Scratch Programming Projects. *Informatics in Education*, 19(4), 611–640. DOI: 10.15388/infedu.2020.27
- Kabátová, M., Kalaš, I., Tomcsányiová, M. (2016). Programming in Slovak Primary Schools. *Olympiads in Informatics*, 10, 125–159.
- Kanemune, S., Shirai, S., Tani, S. (2017). Informatics and Programming Education at Primary and Secondary Schools in Japan. *Olympiads in Informatics*, 11, 143–150.
- Panskyi, T., Rowinska, Z. (2021). A Holistic Digital Game-Based Learning Approach to Out-of-School Primary Programming Education. *Informatics in Education*, 20(2), 255–276. DOI: 10.15388/infedu.2021.12
- Pluhar, Z. (2021). Extending computational thinking activities. *Olympiads in Informatics*, 15, 83–89.
- Pozdniakov, S., Dagienė, V. (eds.) (2019). Informatics in schools. New ideas in school informatics. *ISSEP 2019. Lecture Notes in Computer Science*, vol 11913. Springer, Cham.
https://doi.org/10.1007/978-3-030-33759-9_7
- Tsvetkova, M.S., Kiryukhin V.M. (2021). Algorithmic thinking and new digital literacy. *Olympiads in Informatics*, 15, 105–118.
- van der Vegt, W. (2016). Bridging the gap between Bebras and Olympiad; experiences from the Netherlands. *Olympiads in Informatics*, 10, 223–230.



M. Dolinsky is a lecturer in Gomel State University “Fr. Skoryna” from 1993. Since 1999 he is a leading developer of the educational site of the University (dl.gsu.by). Since 1997 he is heading preparation of the scholars in Gomel to participate in programming contests and Olympiad in informatics. He was a deputy leader of the team of Belarus for IOI’2006, IOI’2007, IOI’2008 and IOI’2009. His PhD is devoted to the tools for digital system design. His current research is in teaching Computer Science and Mathematics from early age.

Change Management in Preparing Indonesian Team to IOI

Felix JINGGA¹, Yugo K. ISAL², Andreas CENDRANATA¹,
Inggriani LIEM³, Adi MULYANTO⁴

¹*Ikatan Alumni Tim Olimpiade Komputer Indonesia (IA-TOKI), Indonesia*

²*Faculty of Computer Science, University of Indonesia, Indonesia*

³*Bebras Indonesia National Board Organization, Indonesia*

⁴*School of Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia*
e-mail: felix@ia-toki.org, yugo@cs.ui.ac.id, andreasc002@gmail.com,
ingebabras@gmail.com, adi@informatika.org

Abstract. Indonesia has been participating at the International Olympiad in Informatics (IOI) since 1995. The process of selecting the four contestants remains unchanged. All medalists in the national level competition were invited to take a series of training camps to select competitively the candidates. In the first two decades of participation in IOI, Indonesia have collected two gold medals, and these surprising achievements were merely more because of luck, and not as a result of a reliable training/selection process. We believe reliable and good training is a basis to predict and expect a stable achievement. Through careful observations and reflections, a change of course of action is needed to bring Indonesia's achievement to the next level. This paper shares Indonesia's experience in changing and implementing strategies during the national training camps to better prepare the four contestants to IOI. What are the changes taken by the coaching team that have been exercised that made it possible for Indonesia to achieve three gold in the last four consecutive years?

Keywords: informatics, olympiad, training, national report, secondary education.

1. Background

Indonesia began to participate in the International Olympiad of Informatics (IOI) in 1995 (Kurnia & Marshal, 2010; Isal *et al.*, 2014), represented by one contestant who won a silver medal. This surprising achievement was covered and echoed by many news media and gained support from many parties to continue Indonesia's participation in future IOIs. The government recognizes IOI as a very prestigious competition which puts the country's reputation at stake, and hence deserves endorsement and support.

The quality of training process is crucial to select the best four contestants. Despite many challenges, since 1995, Indonesia continued to participate in IOI. In the early

years, the selection process was conducted simply by inviting students through school to take the selection process, and then followed by a series of crash training camps. As time went by, beginning in 2002, a more proper and formal competition was conducted nationwide for several international olympiads in science – including informatics – in the form of Olimpiade Sains Nasional (OSN). This annual event is supported by the Ministry of Education and Culture (MoEC) of the Republic of Indonesia and has attracted more participating students. Through a lot of qualifications, there will be around 100 students attending OSN and only 30 students get medals.

Every year, all the OSN medalists of that year are invited to take part in the selection process which consists of four national training camps (Liem, 2016), as shown in Fig. 1. In P1, X represents the number of students who participated in previous year's P1 and P2 that are not qualified to the next stage but still eligible for this year's IOI. In P2 and P3, Y and Z represent the number of students who participated in previous year's P3 and P4 respectively but are still eligible for this year's IOI. Each training camp takes two to three weeks and is held in one university which can provide all the necessary facilities for the training. In each training camp, a set of tasks and various activities are prepared to be done by all the participants. All these activities are designed to select the best four participants who meet some criteria in a competitive and objective manner. The tasks and activities will be described and discussed in detail later in this paper.

Hosting a big international event such as IOI surely requires strong intention, resources and huge concerted efforts to make it happen successfully. However, the experience and lessons learned from its complexity usually gives a “jump/leap” in improvements in many aspects, including the achievement. After its participation in IOI for more than two decades, it was about time for Indonesia to offer and dare to become the host of IOI. The first bidding was proposed in the International Committee (IC) Meeting of IOI 2017 in Teheran, but it was unsuccessful. In the following year, with more determination and preparation, our proposal to become the host of IOI was granted in the IC Meeting and was announced in the last General Assembly Meeting of IOI 2018 in Tsukuba.

Shortly upon returning from Tsukuba, some preparation as the future host of IOI took place with two objectives: success as the host and success to get a higher level of achievement. Until 2018, one silver and one bronze are achievable each year. Table 1 shows the achievement of medals during past IOIs, grouped by interval of three years. See also from Table 1 the fact that since 2016, we managed to get one medal for each contestant, indicated by 0 in the “No Medal” column. This stable achievement reflects the effectiveness of the national training camps (Liem, 2016).

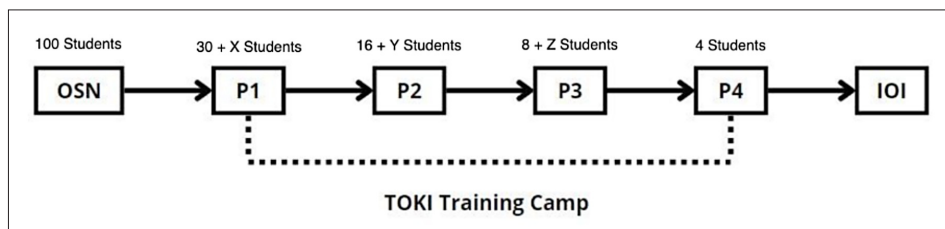


Fig. 1. Preparation Stages to Select the Indonesian Team to IOI.

Table 1
Indonesian Participant Medal Achievements at IOI 1995–2022 (grouped)

Year	Gold	Silver	Bronze	No Medal
2022 (host)	0	3	5	0
2019–2021	3	6	3	0
2016–2018	0	4	8	0
2013–2015	0	4	7	1
2010–2012	0	3	6	3
2007–2009	1	2	8	1
2004–2006	0	5	2	6
2001–2003	0	1	2	5
1998–2000	0	2	3	7
1995–1997	1	1	1	7

How can getting a gold each year be feasible? The two golds that have been collected in the previous IOIs were believed more as luck than a result of a reliable training process. With “GO GET GOLDS” as our motto, several changes in strategies to conduct training camps were made and exercised. In short, as shown in Table 1, one gold was collected each year during 2019–2022 for three consecutive years.

In summary, in the last four years, Indonesia achieved its best results so far in IOI with 3 gold medals, 9 silver medals, 8 bronze medals. These results are new achievement records for Indonesia, and kicked off the “Golden Era of Indonesia” – cemented Indonesia to be a top performing country¹ in the IOI which can be seen in Fig. 2. The aim of this study is to reflect on the changes made during this period for better reshaping

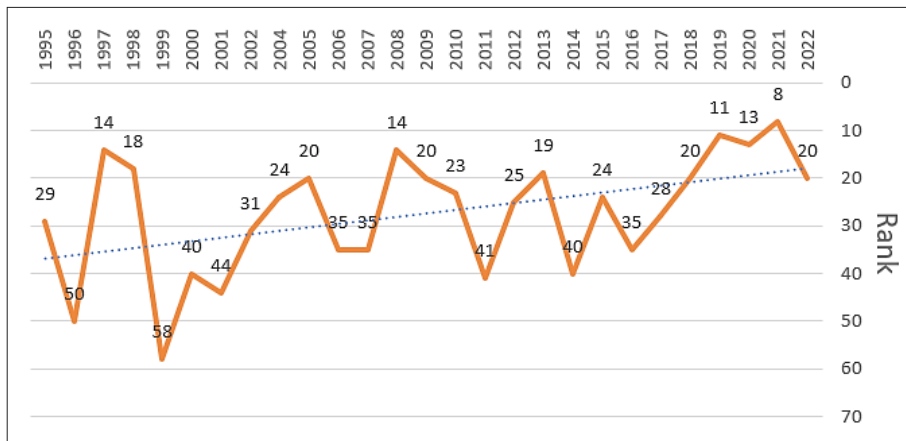


Fig. 2. Plot of Indonesia's Country Ranking in IOI from year to year.

¹ https://stats.ioinformatics.org/delegations/2019?sort=medals_desc
https://stats.ioinformatics.org/delegations/2020?sort=medals_desc
https://stats.ioinformatics.org/delegations/2021?sort=medals_desc

ing of Indonesia's national training program so that stable gold medals can be achieved. Over the years, the overall achievement tends to improve slightly as shown by the dotted line in Fig. 2.

Despite continuous improvement, several questions for further improvement have to be answered by TOKI, especially by the coaching team during the four national training camps. The rest of the paper is organized as follows. Section 2 describes several noticeable challenges to be considered. Section 3 elaborates and discusses the recent changes in Indonesia's TOKI Training Camps. Next, future works and potential challenges are planned and anticipated in Section 4. Finally, Section 5 concludes the discussion by listing all the good practices exercised and good results observed, and implementing them in the future rounds of national training camps.

2. Past Challenges (from 2019 Onwards)

Given the above problem's background, the coaching team needs to identify several challenges they might face, before deciding and making any changes. The following are the list of the challenges under consideration.

2.1. Covid-19

In the wake of COVID-19, discovering a correct formula to run every important event from the lowest selection in school level until reaching the last TOKI Training Camp was quite challenging. There were a lot of uncertainties and limitations that must be followed during this pandemic outbreak. Reduced overall budget from the Government, and regulations that limit people's mobility just to name a few. Even though an online meeting was always possible, there were some IOI candidate students who had internet connection problems. This made the early stage of the TOKI Training Camp not optimal. Having online meetings as the only option, the interaction between IOI candidate students, coaching team, and the board of supervisors become less and not optimal.

2.2. Schedule

The coaching team wants to make the best schedule so that during every TOKI Training Camp, each IOI candidate student can focus only on the training. However, it is hard to satisfy every IOI candidate student's schedule from many schools. During this pandemic, some of the IOI candidate students also get more homework than usual. Some of the students need to attend A-level, university interviews, and some are still deciding which university they want to go to. Some of their schools also do not give them enough excuses so they must attend the exam – even during the TOKI Training Camp contest.

2.3. Skill-gap

Skills gap between the top performer IOI candidate students and the bottom ones is pretty wide. It is inherently hard to make a problem set challenging for the top performer, but still possible to solve for the bottom one. The coaching team considers separating the top performer from the rest, but it is very challenging to maintain balance for both problem sets in the operational level.

2.4. Regeneration

Regeneration for a good Scientific Committee (SC) is quite challenging. The coaching team needs to find SC that is committed to supporting TOKI Training Camps to succeed, considering that they also have their own business to take care of. An SC member should have strong integrity that he/she should avoid conflict of interest, especially when they have a close relation with any IOI candidate students. An SC member is also required to have a certain level of knowledge about IOI. Furthermore, not only the regeneration of SC is important; the regeneration of the candidate students is equally important. We do not want to make sure that we get a great result in one year but get nothing in the following year.

2.5. Synchronizing Sparring Schedules with other Countries

Synchronizing sparring schedules with other countries is also tricky since each country has their own activity with their own schedule. Fortunately, the sparring partners that we have so far are those countries with at most one hour time difference.

3. Changes in Indonesia's TOKI Training Camp

After defining the challenges, the coaching team urged several changes for significant improvement in achievement in the IOI (i.e. getting at least a gold medal in each year) towards hosting the IOI in 2022. The coaching team did some brainstorming and decided to try the new coaching framework. The difference between the new coaching framework and the past will be described in the following subsections.

3.1. Restructuring Coaching Team

Previously, the coaching team structure was always the same as presented by Isal *et al.* (2014). There were four different roles: Participants, (TOKI) alumni, (TOKI bureau)

universities, and Government (Ministry of Education and Culture). Nowadays, a slight change has been made. In this paper, we change the term for Participants to IOI candidate students. As for (TOKI) alumni, it is separated into two parts, which are SC and TC (Technical Committee). Our team structure consists of a Board of Supervisors, a Head Coach, and the SC. The SC is responsible to prepare all the materials and the problem set given in each TOKI Training Camp. During the training camp, SC became the closest one that interacts with the IOI candidate students. Around 10–15 active college students who have experience in IOI or ICPC² volunteer to be in the SC each year. SC used to consist of only TOKI Alumni (Isal, *et al.*, 2014), but now non-TOKI college students who have experience in ICPC are also welcomed. An SC member can reside anywhere in the globe. Meanwhile, the TC members are those who are responsible for the CMS, to ensure that each round of the National Training Camp runs smoothly and effectively (Isal, *et al.*, 2014).

A new role, called Head Coach (HC), is the one who thinks of the training strategy, decides the curriculum, and oversees the whole progress of the National Training Camp. The (TOKI Bureaus) Universities, represented by the Board of Supervisors (SPVs), are those who give full endorsement and ensure that the TOKI Training Camp will be optimally supported by the government, give advice, guidance, and moral support to the candidate students whenever deemed necessary. SPVs members consist of lecturers appointed by the MoEC who have coached in the past since Indonesia's first IOI participation. The Government (MoEC) remains the same.

With the current structure, the HC, SC, and TC will work closely together in coaching the IOI candidate students on a day-to-day basis, while SPVs support the administration needs. Also, HC is responsible for bridging the information between the SC and SPVs. This scenario aligns with the future works quoted in Isal, *et al.* (2014), in which the involvement and contribution of the TOKI Alumni significantly increase (and the involvement of SPVs decrease) in the execution of each round of the National Training Camps.

3.2. Incorporating More Pedagogical Method in National Training Camps

According to Kapur (2020), “Having a well-thought-out pedagogy can bring about improvements in the quality of life of teaching and the way the students can learn.” We feel that the new coaching framework should incorporate more pedagogical methods with careful thinking. The coaching framework is then proposed by borrowing from the following pedagogical concepts.

3.2.1. Goodhart's Law and McNamara Fallacy

According to Strathern (1997), Goodhart's Law is often stated as, “When a measure becomes a target, it ceases to be a good measure.” Fischer (1970) introduced a quanti-

² International Collegiate Programming Contest – the premier global programming competition conducted by and for the world's universities (<https://icpc.global/>).

tative fallacy – called McNamara fallacy – which states that making a decision based solely on quantitative observations (or metrics) and ignoring all other variables often leads to wrong decisions.

Following both the Goodhart's Law and the McNamara Fallacy, we decided not to use a single metric (i.e. points) in determining qualified IOI candidate students in National Training Camps. We also measure outcomes – getting medals in important events. With that being said however, the coaching team track several daily metrics like IOI candidate students' participation in discussions; each IOI candidate student's daily problem solved outside the given contest; how much upsolving they do; how is the performance in each day; etc, only as a signal whether the coaching team need changes or not. This makes the IOI candidate students focus on doing their best in all aspects rather than only when it is in an important contest. We believe that it is important for them to do the practice contest even though its score weight is less than an important contest.

In selecting IOI candidate students that will advance to the next round of TOKI Training Camp, the coaching team also looks into the motivation and behavior aspects. The coaching team do believe that motivation and right behavior can impact each IOI candidate student's performance and growth.

3.2.2. Spacing Effect and Forgetting Curve

According to Ebbinghaus (1885) and estimations from Paul (2007), through spacing effect and repetition, the coaching team can minimize the forgetting effect, as shown by the curve in Fig. 3. Based on this consideration, the coaching team incorporates the repetition and spacing effect to our training method by setting the number of repetitions and their schedule.

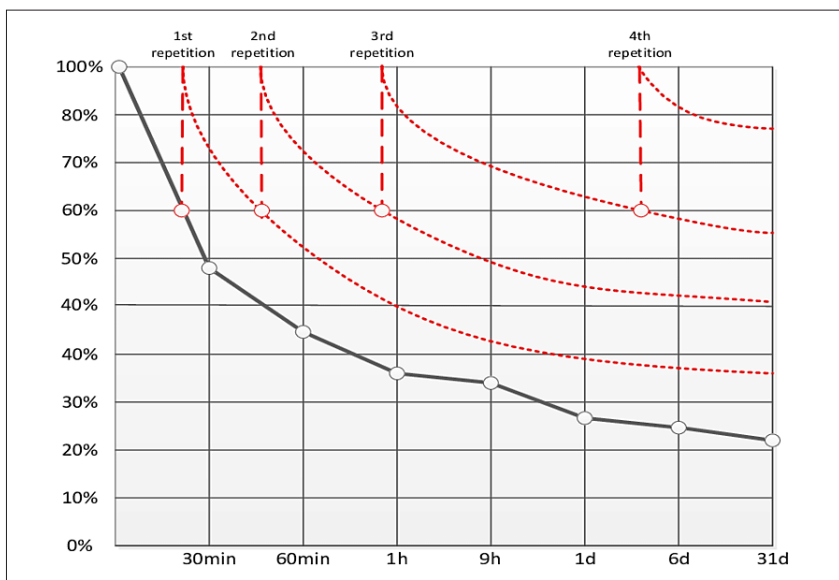


Fig. 3. Alteration of forgetting curve through spacing effect.

Each TOKI Training Camp is usually held in a month period (Liem, 2016). The training activities in the schedule include discussions or knowledge sharings; free time; game time (refer to 3.2.6); practice contests and simulation contests, alternatingly to simulate the spacing effect and repetition. The coaching team encourages IOI candidate students to solve past unsolved problems during the free time session, but it is okay for them to rest if they need to.

3.2.3. *Co-Coaching and Community of Practice*

The coaching team essentially have made the TOKI Training Camp a “community of practice” (CoP) for IOI candidate students. In CoP, it is stated by Lave & Wenger (1991) that, “it is through the process of sharing information and experiences with the group, members learn from each other, and have an opportunity to develop personally and professionally”. Research shows that students learn a concept more deeply when they have to teach it to their peers (Fiorella & Mayer, 2013). Thus, we encourage the IOI candidate students to help each other “level up”.

In this community, the coaching team actively tries to create psychological safety so that the IOI candidate students can freely ask and learn from the coach, mentors, or other IOI candidate students. Not to mention that IOI candidate students can also be coached by other IOI candidate students in one way or another. They should feel safe and confident to share the knowledge they have learned with each other because sharing is a way to help master what they share. In that way, the IOI candidate students can enjoy the learning process more in each round of the TOKI Training Camps. Within the community, they also can re-evaluate their understanding of the material when they share it through the feedback and questions from their peers or coach/mentors to see what they are lacking. With this new setup, not only do they improve their hard skills, but also their soft skills (curiosity, communication, public speaking, etc.).

3.2.4. *Learn – Unlearn – Relearn*

The coaching team decided to make all the TOKI Training Camp materials available in advance. This resulted in the IOI candidate students not knowing which specific topic came out in a contest. It gives each of them the opportunity to analyze the problem with every material that has been given to them previously. This spaced retrieval technique encourages the IOI candidate student to understand the problem at hand comprehensively and to know more deeply about the problem addressed by the material. After the contest, IOI candidate students that do well need to share the insight that they get during the contest (refer to 2.2.3) to others through a slide presentation. The action of doing the contest, solving the problems, making the slides, and presenting it hopefully triggers what we called the “learn – unlearn – relearn” process.

The IOI candidate students “learn” the way of solving the problem when they work on the problem set in the contest. After that, they need to “unlearn” what they discovered during the contest when making the slides by revisiting and once again carefully examining the problem set and thinking about the whys and its proofs. After that, the process of “relearn” is done when they finish the slides and present it to their peers.

3.2.5. *Self-fulfilling Prophecy, Goal Setting and Pygmalion Effect*

The coaching team believes that by setting the goal of getting a gold medal, it will eventually yield the dream result. Chandrasegaran & Padmakumari (2018) claimed that “self-fulfilling prophecies have a significant role in the academic domain. The findings of this research support previous studies, that high teacher expectations produce high student achievement and low expectations produce low achievement”. Based on this claim, it is hoped that each IOI candidate student will have full determination to get a gold medal. Our belief is also supported by a psychological phenomenon called Pygmalion effect in which high goal setting will lead to an improved performance to match that expectation (Mitchell & Daniels, 2003). In every meeting with the IOI candidate students, the coaching team always shows their faith in them that they can get the gold medal.

3.2.6. *Emotional Exhaustion (IOI TOKI Students)*

As stated by Wright & Cropanzano (1998), emotional exhaustion is a chronic state of physical and emotional depletion that results from excessive job, personal demands, and/or continuous stress. TOKI Training Camp is a highly competitive environment and the pressure of doing well in a TOKI Training Camp to achieve the “once in a lifetime” opportunity for representing one’s country in IOI can be very stressful for the IOI candidate students. Due to this, the coaching team realizes that emotional exhaustion (or usually called “burnout”) can also happen to our IOI candidate students. Lack of interest in the work being done, decrease in work performance level, feeling of helplessness, and troubled sleeping are said to be examples of the effect of emotional exhaustion (Aamodt, 2016).

In the case of Indonesian IOI candidate students, burnout can be in the form of their losing motivation and even losing ambition to study anymore. To balance that, the coaching team introduces refreshing activities and calls it “game time”. During this “game time”, the coaching team and the IOI candidate students play games, laugh and chat together to relax and release the stress. They choose games that focus on bonding rather than individual results.

3.2.7. *Talk & Care*

The coaching team not only cares about each student’s knowledge, but also about their mental health (in a limited capacity – the coaching team currently does not have a mental health specialist). The coaching team has a weekly talk with the IOI candidate students, but when they feel that there is something that is troubling them, an additional talk for them to share their problem is offered. It can start from one SC member who is the closest with the student and if any further help is needed, SC will escalate to HC. If necessary, the HC will have a one-on-one session with the IOI candidate student. The talk does not really have to be formal; it can be informal like text chat, or voice call. The coaching team tries to make the students feel as safe as possible. Some examples are when the coaching team needs to adjust our schedule to accommodate IOI candidate students that are taking exams, or give some makeup contests later than scheduled due to

a certain student having a school issue or family issue to be taken care of. The coaching team needs them to feel that they are being cared for.

Another aspect is the motivational talk by the HC, also TOKI's board of supervisors, and/or government officials. In this talk, the coaching team reminds the IOI candidate students how they should love and make an impact for the country. Not only that, the coaching team felt that they need to give the IOI candidate students a purpose/a dream on why they pursue this path not only for their country, but also for themselves.

3.3. *Raising the National Olympiad Lower Bound*

Indonesia also raised the lower bound for the OSN. Some of the material that was given in the First Round of TOKI Training Camp (P1) is now on the syllabus of the OSN.

Another effort is that more TOKI's (even ICPC) Alumni are helping to teach students in the many Provincial Training Camps or schools. With these initiatives, OSN results discover more students that are better in raw talent. It is not uncommon now to have 9th grade students become OSN medallists. These efforts are part of an ongoing initiative mentioned in (Kurnia and Marshal, 2010).

3.4. *Shifting More Advanced Material Earlier*

Before 2019, the coaching team would give more advanced material based on the IOI syllabus in the Third Round of TOKI Training Camp (P3) or later. Due to 2.3, we can start the First Round of TOKI Training Camp (P1) with the more advanced material on the IOI syllabus. The coaching team also can give more problems that have a similar difficulty with IOI. They also can introduce material that goes beyond the IOI syllabus that gives a good insight to help IOI candidate students in solving problems with a higher level of difficulty.

3.5. *Jet-lag Training*

In the Fourth Round of TOKI Training Camp (P4) before IOI, the coaching team tries to replicate it as much as they can so that every contest in the Fourth Round of TOKI Training Camp (P4) is like an IOI contest – we call this “Jet-lag training”. They try to replicate the platform that has been used by IOI that year, the tools, and even the time for the contest. Based on (Mohavedi *et al.*, 2007), the coaching team makes our Fourth Round of TOKI Training Camp (P4) feel like a “mini-IOI” with some other countries joining to be our sparring partners. The coaching team hope that with this setup, the IOI candidate students' arousal level in the practice in the Fourth Round of TOKI Training Camp will be similar to that of IOI.

3.6. Multi-countries Sparring

Since 2019, the coaching team has always tried to conduct many sparring contests with other countries like Singapore, Vietnam, Philippines, Malaysia, and Egypt. The reasons for doing the multi-countries sparring are:

1. The sparring contests benefit all the participating countries.
2. The coaching team wants to know how our IOI candidate students perform not only within ourselves, but also with peers from other countries in the contest. For example, the best student in one school does not mean that he/she is also the best student when he/she meets with other best students from other schools. Considering the “Goodhart’s Law”, the relative rank for each student is only known by the coaching team of each country.
3. There are some materials that are more favored in Indonesia’s National Training Camp but not in other countries’ National Training Camp, and vice versa. Now with the sparring contest, our IOI candidate students are more exposed to materials that are more favored in other countries. Thus, this makes our TOKI Training Camp more challenging and rich in content for our IOI candidate students. For example, Geometry is one of the least favored materials in Indonesia, and this makes our students have less interaction with geometry problems. Multi-countries sparring is a way to balance this.

The multi-countries sparring rounds at first happened due to the HC of these countries knowing each other well, and in spontaneous talk they created this initiative. If feasible, increasing the number of participating countries in the contest would contribute to the enrichment of the problem sets.

TLX³, a contest management platform, made by the Technical Team from IA-TOKI gives a huge contribution to enable these multi-countries sparring rounds and simulate the contest to be similar to IOI.

3.7. Significant Contribution by SC

In each TOKI Training Camp, SC provides materials such as videos, articles, slides, and many more in an internal repository, so that IOI candidate students can learn about the material better. SC also prepares problem sets that are not too difficult but also not too easy with the intention to make them feel better. HC also vetted each problem set so that it is up to standard and TOKI Training Camp syllabus.

Even during COVID-19 period where everything needed to be done online, SC worked hard to produce high quality problems and materials to support the TOKI Training Camp effectively. For example, for each contest in the TOKI Training Camp, two to three SC members had to prepare three to four problems. All of these problems were imported to TLX. Without great teamworks and the significant contributions from SC, the

³ TOKI Learning Center – a self-made Online Judge by IA-TOKI Technical Team (<https://tlx.toki.id>).

TOKI Training Camp would not be going smoothly. Having said that, with the diverse experience and background of all SC members, the coaching team is capable of helping the IOI candidate students to reach gold medal level.

3.8. Pre-national Online Training Camp

As mentioned in (Liem, 2016) about how Indonesia conducts a cycle of training and selections of IOI participants, and also the claimed “One year, or more precisely four training camps three weeks each, is not enough to well prepare IOI participants unless we are lucky to find an extraordinary student”, the coaching team felt the need to further improve the process. Instead of “between two successive camps”, students now will be enrolled in mandatory pre-national online training camp before each phase of TOKI Training Camps. These pre-national training camps aim to equip the IOI candidate students with necessary materials beforehand and plenty of time to do exercises for those given materials.

Oftentimes, the coaching team also asked the IOI candidate students to do mandatory online contests such as other countries OI – COCI, JOI, Google Code Jam, USACO, Codeforces, Atcoder if the schedule of such contests does not collide with the training camps. We asked for the IOI candidate students’ usernames in each of those contests so that the coaching team can gather their performance data. These pre-national online training camps act as a warm-up for the upcoming TOKI Training Camps. This also helps the IOI candidate students against the forgetting effects mentioned in 3.2.2.

3.9. Other Small Things

During the actual IOI contest, there exist some contestants who feel disturbed by the room temperature, constantly audible noise, slow response of the grader, etc, and this might distract their concentration or even ruin their mood which is most important during the competition. Contestants will lose their valuable opportunity if their time is spent complaining. Without any warning, once in a while the coaching team deliberately sets the room temperature too cold (or too warm), slowing the grader’s response to the extreme, making constant/sudden loud noises and observing candidate students’ reaction to the uncomfortable situation. This trains the candidate students to be physically and mentally adaptive to certain situations. During the contest, maintaining full concentration is far more important than spending the valuable remaining time for something that is tolerable/negligible.

4. Future Works and Potential Challenges

Despite all of the changes and improvements described above, the coaching team still wants to make further improvements in the future. The first thing that we want to push

is that we would like to go back to onsite TOKI Training Camp. However, because of the experience we have learned during the pandemic era, we would like to keep the pre-national online training camp online. Even though we have made an onsite national training camp in the past, we would have new challenges given the new normal condition after the pandemic.

The other thing is that there are a lot of programming problems in the world. We as the coaching team need to prepare and archive the problems in a good format so that when we need a certain type of problem, we can find it easily. Not only programming problems, we also want to make more advanced materials with more engaging content.

We also would like to have more different countries in our multi-countries sparring so that point 3.3.6 can make more impact. We would like to expand by trying to collaborate with more countries with small differences in our time zone first. We also want to make it more interesting if we can make a joint training camp with these countries. It will be a big challenge since managing a camp with a joint schedule from each country will be difficult.

Finally, every time before the departure of Indonesia's contingent to IOI, the motto "GO GET GOLDS" keeps reminding us that the character "S" in the last word always means PLURAL.

5. Conclusion

We have presented how Indonesia's achievement progresses throughout the years of participation in IOI. A significant leap happened when a gold medal decorated the achievement each year in three consecutive years – we proudly called it as The Golden Era of Indonesia. Several changes have been discussed, implemented, and their effectiveness observed in the previous sections. The determination and timing as to when the changes were begun to be implemented coincides with the time when Indonesia was appointed to become the Host of IOI, resulting in a surprise of getting one gold in each of the following years. Becoming a host surely means inviting many problems, but the extra pressures could also bring significant improvement in many aspects in the country's training ecosystem. This sharing of Indonesia's experience might be relevant to other countries facing similar challenges and inspire them to become future hosts of IOI.

Acknowledgement

Thanks to the Ministry of Education, Culture, Research, and Technology through the National Achievement Center, we are able to conduct National Training Camps. Thanks also to the National Training Camp Board of Supervisors that support many coaching team's decisions. Thanks also to the Technical Team of IA-TOKI led by Ashar Fuadi that

develop and maintain TLX to support all of the training camps. Thanks to members of IA-TOKI and all the volunteers that could not be mentioned in this paper for the direct/indirect contribution for the training camp. Thanks to Kezia Aurelia Cendranata and Dennis Setiawan for helping us in proofreading.

References

- Aamodt, M. (2016). *Industrial/Organizational Psychology: An Applied Approach* (8th ed.). Boston, MA: Cengage Learning. p. 563. ISBN 978-1-305-11842-3.
- Chandrasegaran, J., Padmakumari, P. (2018). <https://files.eric.ed.gov/fulltext/EJ1186415.pdf>
- Ebbinghaus, H. (1885). *Memory: A Contribution to Experimental Psychology*. New York: Dover.
- Fiorella, L., & Mayer, R.E. (2013). The relative benefits of learning by teaching and teaching expectancy. *Contemporary Educational Psychology*, 38(4), 281–288.
<https://doi.org/10.1016/j.cedpsych.2013.06.001>
- Fischer, D.H. (1970). *Historians' fallacies: toward a logic of historical thought*. Harper Torchbooks (1st ed.). New York: HarperCollins. p. 90. ISBN 978-0-06-131545-9. OCLC 185446787.
- Isal, Y.K., Liem, M.M.I., Mulyanto, A., Marshal, B. (2014). Indonesian Olympiad in Informatics: Significant advancements between 2010 and 2014. *Olympiads in Informatics*, 8, 191–198.
- Kapur, R. (2020). *Understanding the Meaning and Significance of Pedagogy*.
- Kurnia, I. & Marshal, B. (2022). *Indonesian Olympiad in Informatics*.
- Lave, J., Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press. ISBN 978-0-521-42374-8.
- Liem, M. (2016). Reshaping Indonesian Students Training for IOI. *Olympiads in Informatics*, 10, 195–205. DOI: 10.15388/ioi.2016.12.
- Mitchell, T., Daniels, D. (2003). *Motivation*. DOI: 10.1002/0471264385.wei1210.
- Movahedi, A., Sheikh, M., Bagherzadeh, F., Hemayattalab, R., Ashayeri, H. (2007). A practice-specificity-based model of arousal for achieving peak performance. *J Mot Behav.*, 39(6), 457–62. DOI: 10.3200/JMBR.39.6.457-462. PMID: 18055352.
- Paul, K. (2007). *Study Smarter; Not Harder*. Self-Counsel Press
- Strathern, M. (1997). 'Improving ratings': audit in the British University system". *European Review*. 5(3), 305–321. DOI: 10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4.



F. Jingga was a faculty member of School of Computer Science, Binus University. He is now the active Head Coach for TOKI, leading TOKI to achieve 3 Golds for several consecutive years from 2019. He is a Certified Professional Coach by ICF (International Coaching Federation).



Y.K. Isal is a faculty member of the Faculty of Computer Science, University of Indonesia. He is an active organizer, coach and judge for Indonesian Olympiad in Informatics since 2006. He loves photography and music, and wrote the lyrics of the IOI Theme Song.



A. Cendranata was one of the 2021 ICPC World Finalists in Dhaka, and also the 2nd Indonesian Team team leader in IOI 2022. He has since helped to coach Indonesia's IOI team from 2019 until now.



I. Liem was a faculty member of School of Electrical and Engineering, Bandung Institute of Technology from 1979 to 2018. She is an active organizer, coach and judge for Indonesian Olympiad in Informatics since 2004. She is the Leader of Bebras Indonesia NBO since 2016.



A. Mulyanto is a faculty member of School of Electrical Engineering and Informatics, Bandung Institute of Technology. He is an active organizer, coach and judge for Indonesian Olympiad in Informatics since 2004. His special role as an organizer is in managing coordination among stakeholders such as universities and the Ministry of Education and Culture.

National Olympiad in Informatics: Sri Lanka

Suvin Nimnaka KODITUWAKKU, Thisari GUNAWARDENA

University of Colombo School of Computing, Sri Lanka
e-mail: hello@suvin.me, thisarigun99@gmail.com

Abstract. The National Olympiad in Informatics (NOI) in Sri Lanka is an annual programming competition for students in Sri Lanka. The competition aims to identify and encourage talented young programmers in the country. The competition consists of several rounds, including training rounds, a screening round, a national round, and a training camp for the top performers. The problems in the competition are designed to test the students' programming skills, algorithmic thinking, and problem-solving abilities. The NOI Sri Lanka competition plays an important role in promoting the development of computer science and programming skills among young people in Sri Lanka.

Keywords: IOI, programing olympiads, informatics, distance learning, informatics.

1. Introduction

The National Olympiad in Informatics (NOI) serves as a significant platform for students in Sri Lanka to cultivate their skills in computer science and programming. The competition aims to promote the development of theoretical knowledge, programming proficiency, and logical, critical, and creative thinking among students. Through the identification of exceptional participants, the competition facilitates the advancement of computer science education, raises awareness among secondary school students and teachers, and ultimately selects a national team to represent Sri Lanka at the International Olympiad in Informatics (IOI).

The National Olympiad in Informatics (NOI) was established in 1992 with the aim of providing a platform for Sri Lankan students to enhance their theoretical knowledge and programming skills in computer science. Despite its initial success, the program faced various challenges in management and organization, which led to stagnation in its progress. However, in 2018, a group of IOI Informatics Olympiad Alumni took the initiative to revive the program. Since then, the organizers of the NOI have been working tirelessly to improve the quality of the program year by year, making it one of the foremost olympiads in Sri Lanka. The Ministry of Education in Sri Lanka has also endorsed the competition, recognizing its importance in promoting computer science education and identifying exceptional students in the field. The competition includes

training rounds, a screening round, a national round, and a training camp for the top performers, with one-on-one discussions available to resolve any issues encountered during the process.

The competition consists of several rounds, including training rounds, a screening round, a national round, and a training camp for the highest-performing participants. In addition, the organizing committee provides opportunities for one-on-one discussions between students and mentors to address any issues encountered during the competition. These initiatives provide a comprehensive platform for students to hone their skills and gain valuable experience.

The remainder of this country report is organized as follows: in Section 2 the organizational structure of the NOI is discussed; in Section 3 the program structure of the NOI is discussed; in Section 4 the performance of the Sri Lankan Delegation at the IOI is discussed; in Section 5 the syllabus followed at the competition is discussed; the Section 6 discusses how the NOI was conducted during the COVID-19 pandemic; the Section 7 concludes the report and Section 8 provides the acknowledgments

2. Organizational Structure of the NOI

The NOI organization is a part of the ACM Student Chapter of UCSC managed under the University of Colombo School of Computing (UCSC). The NOI organization has 3 distinct sections, namely, (i) EAC – The Executive Advisory Committee, (ii) SC – The Scientific Committee, (iii) OC – The Organizing Committee as shown in the Fig. 1.

2.1. Executive Advisory Committee

The EAC is composed of the academic staff members of the University of Colombo School of Computing (UCSC). The Director of the UCSC acts as the Patron of the NOI program and the Faculty Advisor of the ACM Student Chapter of UCSC also

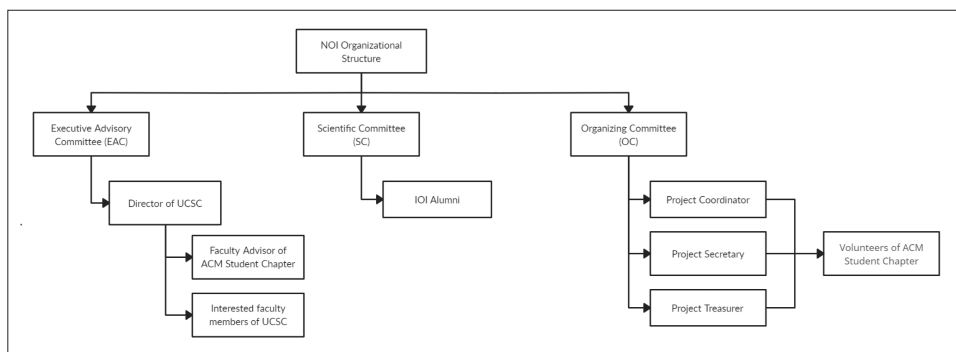


Fig. 1. Organizational Structure of the NOI.

acts as the Faculty Advisor for the NOI program. Additionally, the previous faculty sponsors and staff members of the UCSC who are interested in the program are part of the EAC.

2.2. Organizing Committee

The OC is responsible for organizing the entire NOI program. It includes volunteers from the ACM Student Chapter of UCSC and there are 3 major positions within the OC.

- Project Coordinator – Responsible for the entire NOI program.
- Project Secretary – Responsible for the operational activities.
- Project Treasurer – Responsible for the financial activities.

The coordinator, secretary and treasurer are appointed by the Executive Committee of the ACM Student Chapter of UCSC.

The team leader and the deputy leader for the Sri Lankan delegation for the International Olympiad in Informatics (IOI) are appointed from the Scientific Committee and the Organizing Committee, one from each section. In case the team leader is selected from the SC, the deputy leader will be selected from the OC, and vice versa.

2.3. The University of Colombo School of Computing

In response to the rapid development and evolving nature of the Information Technology field, coupled with anticipated structural changes, there arose an urgent need for a higher education institution dedicated to computing in Sri Lanka. In 2002, Vidya Jyothi Professor V. K. Samaranayake founded the University of Colombo School of Computing (UCSC) with the objective of establishing such an institution.

Since its inception, the UCSC has established a distinguished reputation as the foremost higher education institution for computing in Sri Lanka. Its primary objective is to equip students with the necessary knowledge and skills to pursue careers in Information and Communication Technology, such as Software Developers, Systems Analysts, Network Administrators, Database Administrators, Web Developers, IT Managers, IT Strategic Planners, and IT Policy Makers.

3. Program Structure of the NOI

The NOI Sri Lanka program's structure has 3 stages (as shown in Fig. 2);

- (i) Monthly practice contests.
- (ii) Online qualifier round.
- (iii) The National Olympiad in Informatics.

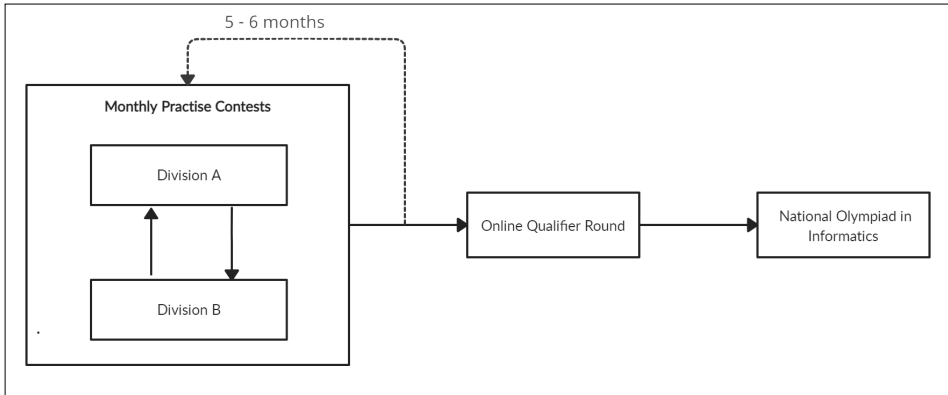


Fig. 2. Program Structure of the NOI.

3.1. Monthly Practise Contests

The NOI program starts with the monthly practice contests. The participating students will be split into Division A and Division B, and separate monthly contests will be hosted for each of them. Students who are interested in participating in the NOI program are required to create an account at <https://www.noi.lk>. Once the registration is completed, the students will have access to the NOI Portal (<https://portal.noi.lk/>), which is the Learning Management System (LMS) of the NOI program. All the announcements, contests and other material related to NOI Sri Lanka will be published on the aforementioned LMS.

3.1.1. Division A

Each year, Division A begins with the eight participants who had the highest scores during the NOI program the year before (Wang *et al.*, 2010). Participants who enter this division will remain until the end of the next Division B monthly practice contest. Every month, the participants will be required to take part in a particular set of programming contests that the Scientific Committee determines. The contests are chosen from CodeForces¹, AtCoder² and CodeChef³. In most cases, there will be 4 contests each month. The score of the participants in the division A program will be calculated based on the score obtained by participating in the aforementioned contests by the Scientific Committee and the participants will be ranked accordingly. At the end of each month, the last ranking two members of Division A will be demoted to Division B.

Contestants in the division A program are allowed to compete in Division B monthly contests, but their results will not be considered for the final leaderboard of the Division B contest.

¹ Available: <https://codeforces.com/>

² Available: <https://atcoder.jp/>

³ Available: <https://www.codechef.com/>

3.1.2. Division B

Apart from the Division A participants, every other participant will be included in Division B. Contestants in this division have to face separate monthly practice contests hosted by the ACM Student Chapter of UCSC in collaboration with the Scientific Committee of the NOI. At the end of each Division B monthly contest, the top two contestants of the Division B program will be promoted to the Division A program.

The monthly contests will continue for about 5 months depending on the timeline of the International Olympiad in Informatics (IOI).

3.2. Online Qualifier Round

The online qualifier round serves as a screening test to select the topmost participants of the large participant pool of Division B of the NOI. The competition is open to all primary and secondary school students in Sri Lanka who are younger than 20 years. The contest will be hosted on HackerRank⁴ by the ACM Student Chapter of UCSC. In previous years, the qualifier round was held as a 5-hour contest where the participants were asked to be present at the UCSC. However, due to logistical challenges and travel challenges, the competition was switched to a virtual mode. During the COVID-19 pandemic, the online qualifier was held as a 5-hour online contest with virtual proctoring but later in 2022, the contest was switched to a 12-hour virtual contest where the participants were given 6 problems to solve.

The student submissions were manually checked for plagiarism by the Scientific Committee and volunteers from the ACM Student Chapter of UCSC. Only the students with at least one valid submission were issued a certificate of participation. The top 8 participants with the highest scores were selected to compete in the National Olympiad in Informatics contest (also known as the final round). In addition to the 8 participants from the qualifier round, the 8 students who remained in Division A after the final practice contest will be competing in the National Olympiad in Informatics. Altogether, 16 students will be participating in the National Olympiad in Informatics.

3.3. The National Olympiad in Informatics

The National Olympiad in Informatics is a similar contest to the International Olympiad in Informatics (IOI). The contest will run for 2 days and contestants will receive 3 problems to solve each day within a 5-hour time block. As similar to the previous contests, the problems will be set by the Scientific Committee in collaboration with the Informatics Olympiad alumni. The 4 students who score the highest marks in this competition and who satisfy the eligibility criteria of the IOI will be representing Sri Lanka at the International Olympiad in Informatics (IOI).

⁴ Available: <https://www.hackerrank.com/>

3.3.1. Awards

Considering the scores of the National Olympiad in Informatics, the following awards will be awarded. Unlike the competition, the medals are awarded according to two age groups to motivate and increase the participation of younger students.

- The IOI Sri Lankan delegation – Awarded to the 4 students who score the highest in the National Olympiad in Informatics and get selected to the Sri Lankan delegation for the International Olympiad in Informatics (IOI).
- Under 16 age category:
 - Gold – Awarded to the student with the highest score in the under-16 age category.
 - Silver – Awarded to the student with the second-highest score in the under-16 age category.
 - Bronze – Awarded to the student with the third highest score in the under-16 age category.
- Under 20 age category:
 - Gold – Awarded to the student with the highest score in the under-20 age category.
 - Silver – Awarded to the student with the second-highest score in the under 20 age category.
 - Bronze – Awarded to the student with the third highest score in the under-20 age category.
- Special Awards:
 - Best Performing Contestant – Awarded to the student with the highest score regardless of the age category.
 - Best Performing School – Awarded to the school that has the highest count of students among the finalists who participated in the final round, and have achieved a score that is greater than the median.
 - Best Performing Girl Coder – Awarded to the female participant with the highest score regardless of the age category to encourage female participants to take part in the contest.

The awards ceremony will be held after the Sri Lankan delegation returns home from the International Olympiad in Informatics (IOI).

4. Performance of the Sri Lankan Delegation

The following section will discuss the performance of the Sri Lankan delegation at the International Olympiad in Informatics (IOI) over the past 5 years.

According to the performance statistics from the International Olympiad in Informatics-Statistics⁵, after the revival of the competition in 2018, a steady improvement could be observed in the IOI Team Performance. The IOI Team Performance shown

⁵ Available: <https://stats.ioinformatics.org/>

in Fig. 4 is calculated by averaging the sum of each contestant's average marks as follows.

$$\text{contestant_average} = \frac{\text{marks_obtained_by_contestant}}{\text{total_marks}} * 100\%$$

$$\text{IOI_team_performance} = \frac{\sum \text{contestant_average}}{\text{number of contestants}}$$

It could be observed from Fig. 3 that the IOI Team Performance declined with the collapse of the NOI but after the revival of the program, the team is observed to be making steady progress as depicted in Fig. 4. Also, it could be observed as shown in Fig. 4,

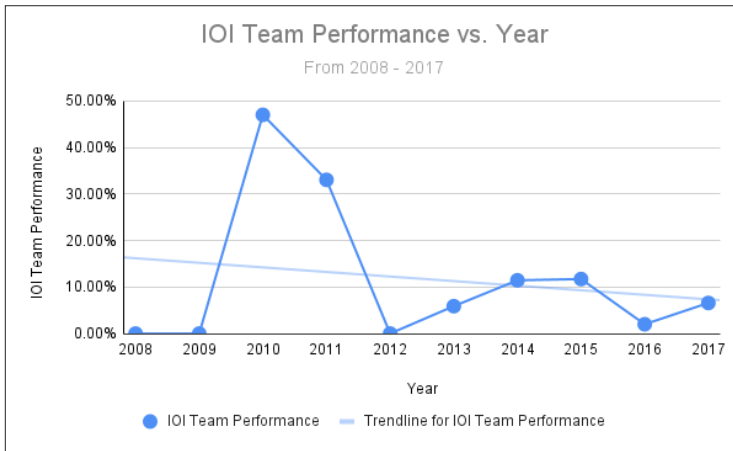


Fig. 3. IOI Sri Lanka Team Performance from years 2008 to 2017.

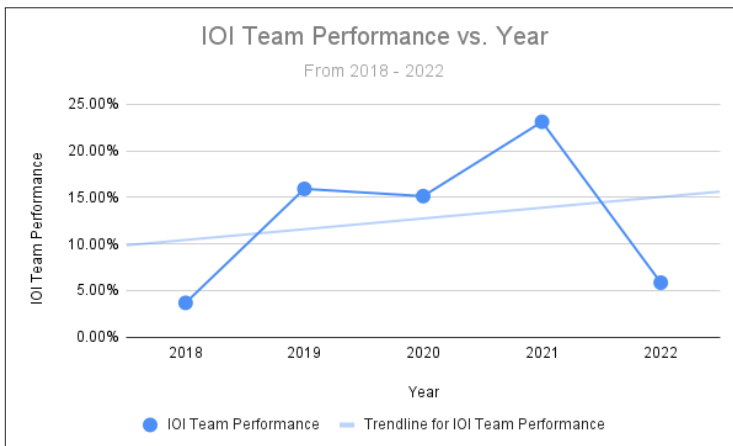


Fig. 4. IOI Sri Lanka Team Performance from years 2018 to 2022.

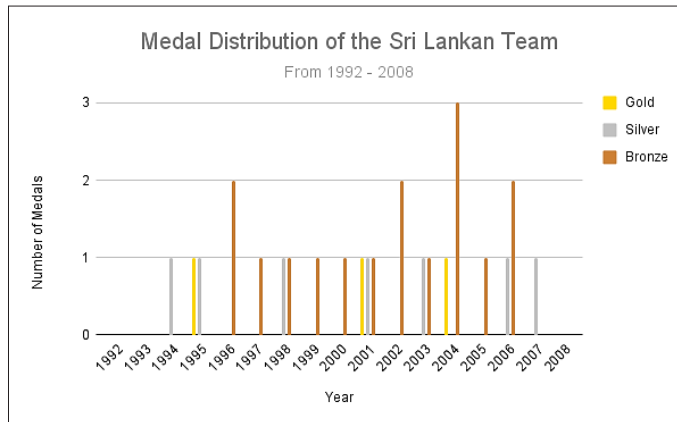


Fig. 5. Medal Distribution of the Sri Lankan Team.

the Sri Lankan Team saw a commendable improvement in IOI 2021 but it declined drastically in IOI 2022. The Scientific Committee is working on identifying probable causes and taking measures to improve the performance of the team in IOI 2023.

As seen in Fig. 5 from 1994 up until 2007, the Sri Lankan team has been able to win medals consistently⁶. However, due to the collapse of the program, for nearly 10 years, the team was not able to win any medals. After the revival of the program in 2018, the team came close to winning a medal but missed by just 5 points in IOI 2021⁷.

5. Syllabus and the Tasks

A syllabus is important as a guide for the problem setters of NOI and for making sure that the relevant competencies expected of the contestants are covered properly. The syllabus of each year's NOI program is compiled at the beginning of the program by the aforementioned Scientific Committee to make sure all the competencies are covered at the monthly practice contests, online qualifier round and the NOI. It is compiled by considering the IOI Syllabus⁸ and past IOI questions⁹.

The following outlines the syllabus that was used in the National Olympiad in Informatics 2022:

- Graphs and Trees
 - Shortest Path
 - Connected Component/ Disjoint Set
 - Directed Acyclic Graph
 - Minimum Spanning Tree

⁶ Available: <https://stats.ioinformatics.org/results/LKA>

⁷ Available: <https://stats.ioinformatics.org/results/2021>

⁸ Available: <https://ioinformatics.org/page/syllabus/12>

⁹ Available: <https://ioinformatics.org/page/contests/10>

- Algorithmic Strategies
- Greedy Algorithm
 - Brute Force Algorithm/ Implementation
 - Divide and Conquer
 - Dynamic Programming
 - Recursion
 - Backtracking
- Binary Search
- Two Pointers
- Ad-hoc
- String Algorithm
 - Substring
- Bitmask/Boolean algebra
- Combinatorics
 - Inclusion/exclusion
 - Pigeon Hole
 - Pascal's Identity
 - Binomial Theorem/Coefficients
- Data Structures
 - Binary Indexed Tree
 - Segment Tree
 - Stacks
 - Heap
 - Trie
- Miscellaneous:
 - Covers any other topic that is not included in the aforementioned syllabus

Throughout the program, all the above were covered in varying difficulty levels from easy, medium to hard as seen fit by the Scientific Committee.

5.1. Sample Tasks

The sample tasks can be demonstrated under 3 categories considering the program structure of the NOI program; (i) Monthly contests (ii) Qualifier round (iii) NOI final round.

5.2. Monthly Contest Problems

COVID-21 Variant: The problem involves analyzing interactions between people in a room to determine if there is at least one person who, if infected, would spread the disease to the entire room. The interactions are recorded without timestamps. The spreading behavior is influenced by age and distance. The goal is to determine if COVID-19 and COVID-21 variants would spread to the whole room based on the given interactions. The input consists of the number of test cases, the number of people, the number of

interactions, the age of each person, and the distances between interacting people. The output should indicate whether COVID-19 and COVID-21 would spread to the whole room for each test case (Armoni *et al.*, 2006).

33.33% of contestants who attempted this question obtained full marks.

Topic(s): Graph (MST), Contest: Monthly contest – April 2021.

Palindromic Substring Discovery: The task is to find the largest substring within a given string that can be rearranged to form a palindrome. The goal is to determine the maximum length of such a substring.

33.33% of the contestants who attempted obtained full marks for this problem.

Topic(s): Bitmask, Dynamic Programming, Contest: Monthly contest – January 2021.

5.3. Qualifier Round Problems

Petrol Queue: The task is to rearrange the distribution of items in an array in non-increasing order. The input consists of an array representing the initial distribution, and the goal is to determine the minimum number of operations required to achieve a non-increasing order. At each operation, two adjacent positions can be selected to perform a transformation. The possible transformations involve removing an item from one position and adding it to the adjacent position, or vice versa. The output should be a single integer representing the minimum number of operations needed.

A comprehensive solution has not been proposed by the candidates, and the maximum score achieved was 81.08%.

Topic(s): Dynamic Programming, Contest: Qualifier round 2022.

Building the artifact: The task is to determine if it is possible to build an artifact by collecting interconnected parts. The input consists of the number of parts and their connections, where each connection indicates that one part must be collected before another. However, some parts may have a self-connection, making them impossible to collect. The goal is to output “YES” if it is possible to build the artifact, and “NO” if it is not possible.

6.98% of the contestants who attempted have obtained full marks.

Topic(s): Topological sort, Contest: Qualifier round 2021.

5.4. Final Round Problems

Mouse in the Exploding Maze: The task is to calculate the minimum time to traverse a 2D grid (R rows x C columns) from a starting cell to an exit cell. The grid contains free cells, blocked cells, or cells with a toggling obstacle that alternates between accessible and inaccessible every minute. Movement is allowed to an adjacent cell (up, right, down, left) or to remain stationary each minute. The starting and exit cells are always free.

4.17% of the contestants who attempted obtained full marks for this problem.

Topic(s): Graph traversal, Contest: Final round 2019.

Eat Your Peanuts: The task is to assign N contiguous segments, each containing a specific quantity, among a set of F recipients in a way that minimizes the maximum total quantity allocated to any single recipient. Each recipient should be given one or more adjacent segments, and all segments must be assigned. The program should output a single integer, representing the maximum total quantity received by any single recipient under the optimal assignment.

A comprehensive solution has not been proposed by the candidates, and the maximum score achieved was 54.02%.

Topic(s): Binary search, Contest: Final round 2019.

Find the Worlds: The task is to analyze a set of points representing cities in a 2D universe, with each city located at a unique integer coordinate (x, y) . Cities are grouped into distinct worlds such that the euclidean distance between any two cities within a single world is $\leq 10,000$, and the distance between any two cities from different worlds is $> 10,000$. The program should determine and output the number of worlds and the number of cities within each world.

A comprehensive solution has not been proposed by the candidates, and the maximum score achieved was 95%.

Topic(s): Ad-hoc, Hashing, Contest: Final round 2020.

6. Effects of the COVID-19 Pandemic

The COVID-19 pandemic has resulted in significant disruptions to educational institutions and their operations worldwide. Social distancing and other public health measures necessitated the closure or transition to remote learning for many schools and universities. The National Olympiad in Informatics has been affected by the pandemic in terms of logistics, organization, student participation, and performance. In light of all the issues associated with public health concerns, the Organizing Committee has made the decision to conduct the contest entirely online (Dhawan, 2020).

6.1. Early Efforts

During the initial stages of the COVID-19 pandemic, the National Olympiad in Informatics (NOI) related contests were conducted virtually using Zoom¹⁰ video conferencing software for supervision. The participants were grouped and assigned a proctor, who was typically a volunteer recruited from the University of Colombo School of Computing (UCSC). Each group was allocated a Zoom breakout room and instructed to activate their web cameras, enabling the proctor to monitor their participation.

Despite the Organizing Committee's efforts, instances of cheating by contestants were discovered during the virtual National Olympiad in Informatics (Arkorful et

¹⁰Available: <https://www.zoom.us/>

al., 2015). Moreover, other issues such as a lack of computer devices resulting from low family income and inadequate connectivity infrastructure in rural areas affected some students. In response to these challenges, the OC devised an innovative solution aimed at ensuring the contest's integrity and ensuring inclusivity across diverse student backgrounds.

6.2. *The NOI Virtual Proctor*

The NOI Virtual Proctor is a simple web-based tool where it provides students with a safe exam environment to participate in the National Olympiad in Informatics contests. The students were onboarded into the platform with their registration details and they were informed of their respective credentials before the contest.

National Olympiad in Informatics Virtual Proctor required initial permission to access the web camera and to screen record in order to capture the student's environment during the initial sign-in. After unlocking the contest at the start time, the tool monitored the student's browser tabs to verify that only authorized pages were accessed. The allowed web pages were pre-configured into the platform by the OC, and in this instance, only the C++ manual and the Hackerrank¹¹ contest were permitted.

In the event of an unauthorized tab being opened, no warning was given to the student, but the incident was immediately reported to a Slack¹² channel, which contest officials monitored throughout the contest. The tool periodically captured the candidate's screen and uploaded it to cloud storage in the form of 2-minute chunks. In cases of connectivity issues, the tool cached the video chunks and incident reports on the local machine and synced them with the cloud once the connection was re-established.

National Olympiad in Informatics Virtual Proctor significantly aided in the management of the contest and in minimizing attempts of cheating during the NOI.

The same group of organizers who developed the National Olympiad in Informatics Virtual Proctor tool further developed it into a commercial-grade product capable of accommodating online evaluations on a larger scale, globally. Additional information on this tool can be found via the following link: <https://proktara.com>. This development serves as a testament to the innovation and adaptability of the NOI Organizing Committee in response to the challenges presented by the COVID-19 pandemic.

Students who faced difficulties in obtaining the necessary equipment to compete in the NOI were provided with an online form to inform contest officials of their situation. The OC took necessary measures to accommodate the students who responded, including providing equipment such as laptops and internet routers. Volunteers from the UCSC were dispatched to the students' locations with the necessary equipment and to monitor them throughout the contest, while strictly adhering to the health guidelines imposed by the Sri Lankan government. This initiative ensured that all students, regardless of their backgrounds, were given equal opportunities to participate in the contest.

¹¹ Available: <https://www.hackerrank.com/>

¹² Available: <https://slack.com/>

7. Conclusion

It could be observed that after the revival of the National Olympiad in Informatics competition in 2018, the quality of the competition has been improving year by year. Unprecedented challenges such as the COVID-19 pandemic and financial instabilities in Sri Lanka did not hinder the progress of the competition. Identifying strengths and weaknesses after each year's competition and taking a course of action that further increases the strengths and overcomes weaknesses has been vital in increasing the impact of the competition.

In conclusion, the NOI competition can have a significant impact on the education system and students of Sri Lanka. It provides an opportunity for exceptional students to showcase their programming skills and compete with peers from all around the country. Garcia-Mateos (Garcia-Mateos and Fernandez-Aleman, 2009) and Dagienė (Dagienė, 2010) have noted the influence of programming, and informatics olympiads in studying computer science. Thereby, participating in the NOI can inspire students to pursue higher education and professional careers in Computer Science and related disciplines, consequently raising the standard of Computer Science education in Sri Lanka.

8. Acknowledgements

We would like to express our gratitude to all those who have contributed to the successful completion of the National Olympiad in Informatics (NOI). First and foremost, we would like to thank the Ministry of Education – Sri Lanka and the University of Colombo School of Computing (UCSC) for unwavering support and guidance throughout the entire NOI program.

We would also like to extend our sincere gratitude to the ACM Student Chapter of University of Colombo School of Computing and all the volunteers who generously gave their time and effort to the NOI program. Without their willingness to share their talents and expertise, the NOI program would not have been a success.

Furthermore, we are grateful to the Informatics Olympiad alumni, our colleagues and our friends who provided us with their encouragement and assistance in various aspects of the NOI. Their contributions, no matter how small, have been significant in helping us navigate the challenges of conducting a successful NOI program.

Lastly, we acknowledge the financial support provided by all the institutes, which enabled us to carry out the NOI. We are deeply appreciative of their investment in this program and the opportunities it has provided us.

We recognize that there are many others who have contributed to the National Olympiad in Informatics in one way or another, and we apologize for any unintentional omission. Nevertheless, we are grateful for everyone who has supported us in this endeavor.

References

- Arkorful, V., Abaidoo, N. *et al.* (2015). The role of e-learning, advantages and disadvantages of its adoption in higher education. *International Journal of Instructional Technology and Distance Learning*, 12(1), 29–42.
- Armoni, M., Gal-Ezer, J., Hazzan, O. (2006). Reductive thinking in computer science. *Computer Science Education*, 16, 281–301.
- Dagienė, V. (2010). Sustaining informatics education by contests. In: *Teaching Fundamentals Concepts of Informatics: 4th International Conference on Informatics in Secondary Schools-Evolution and Perspectives, ISSEP 2010, Zurich, Switzerland, January 13–15, 2010. Proceedings 4*. pp. 1–12.
- Dhawan, S. (2020). Online learning: A panacea in the time of COVID-19 crisis. *Journal of Educational Technology Systems*, 49(1), 5–22.
- Garcia-Mateos, G., Fernandez-Aleman, J.L. (2009). Make learning fun with programming contests. *Transactions on Edutainment II*, pp. 246–257.
- Wang, H., Yin, B., Liu, R., Tang, W., Hu, W. (2010). Selection mechanism and task creation of Chinese national olympiad in informatics. *Olympiads in Informatics*, 4, 142–150.



S.N. Kodituwakku graduated from the Department of Information Systems Engineering of the University of Colombo School of Computing, Sri Lanka. He headed the National Olympiad in Informatics in the years 2021–2022. He was the deputy leader for the Sri Lankan delegation at the IOI in 2021 and the team leader in 2022. He is also a former scientific & technical committee member of the International School of Informatics for Juniors. His current research is in Human Computer Interaction, focusing on visual analytics.



T. Gunawardena is an undergraduate of the Department of Computation and Intelligent Systems, University of Colombo School of Computing, Sri Lanka. She served as the secretary of the National Olympiad in Informatics (NOI) of Sri Lanka from 2021–2022. She is a member of both the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). Her current research interest lies in emotion-based music generation using deep learning techniques.

Team Competition in Informatics and Mathematics “Cēsis”

Mārtiņš OPMANIS¹, Diāna SILIŅA², Sandra SILIŅA¹,
Pēteris PAKALNS

¹*Institute of Mathematics and Computer Science, University of Latvia,*

²*Cēsis State Gymnasium*

*e-mail: martins.opmanis@lumii.lv, diana.silina@inbox.lv, sandra.silina@lumii.lv,
peterispakalns@gmail.com*

Abstract. Team competition in informatics and mathematics for high-school students “Cēsis” still is a rare competition where usage of Internet resources is allowed. Internet availability puts additional requirements for competition tasks. We give a brief overview of used task types together with task examples. Fast and accurate evaluation and grading of solutions to non-standard tasks in a limited time interval is challenging, and automated evaluation tools used at the competition are described.

Keywords: team competition, automated evaluation, Internet resources, task types.

1. Introduction

In the third volume of “Olympiads in Informatics,” there was a paper about the team competition in informatics and mathematics (TCIM) “Ugāle” (Opmanis, 2009). TCIM “Ugāle” was an annual event organized until its 20th edition in 2015. Since 2016 competitions almost in the same format have been organized by Cēsis State Gymnasium. In the following, by the abbreviation TCIM, the TCIM “Cēsis” will be assumed. Relocation of onsite competition from one town to another, even if problem setters remain the same, is not a trivial task, and some requirements and observations from the experience together with the general competition format are described in the second section, “From Ugāle to Cēsis.” The general structure of the competition is described in the third section, while tasks and solutions of various task types are described in the fourth section. While the main features of the competition were kept traditional, there are essential differences, including the evaluation and grading of solutions described in the fifth section. The last section briefly mentions features of the new evaluation and grading system, “UIM.”

2. From Ugāle to Cēsis

At the end of 2014, in discussions with Ugāle Secondary School teacher Aivars Žogla, the main organizer of TCIM “Ugāle,” it became clear that it would be better to finish organizing competitions in Ugāle since TCIM here was organized for 20 years, and in some sense reached its limits. At least for local organizers, onsite finals became a routine task for a few passionate people who established this event long ago. Not the slightest reason for this decision was almost a predefined impossibility for host teams to successfully compete with the strongest teams from the gymnasiums from the capital of Latvia, Riga, and other cities.

It should be noted that almost any event has its beginning and end. Even with the same title, the content may be completely different. The biggest challenge in our case in 2015 was providing competition in the usual way, already knowing that this would be the last one in the current form and Ugāle. Since there were people ready to continue contributing to organizing similar events even if the competition would be relocated, the process to find another host – an organization and, most importantly, passionate people willing to do this, started. It was clear that the new host should also be outside Riga. After rejecting some mostly theoretical candidates, there came a proposal from Cēsis State Gymnasium and namely teachers Diāna Siliņa and Agrita Bartušēvica to host the competition in Cēsis, one of the eldest towns (established in 1206) in Latvia. While towns are located in different regions (Ugāle in Kurzeme, Cēsis in Vidzeme), its distance from Riga (95 km), traditionally having most finalists, makes it a perfect place for the event.

3. The General Format of TCIM

TCIM is an annual high-school team competition provided in two rounds – supervised online semifinals in January and onsite finals in May. Each team consists of three participants and may use one computer with access to the Internet. Due to the COVID pandemic in 2020, only the semifinals were organized, and the 2021 competition was canceled altogether. Up to now, seven semifinals and six finals have been organized.

In the semifinals, there is one week when the local supervisor should find a 5-hour window for a local contest. All solutions are sent for evaluation to TCIM organizers via e-mail. After evaluation, finalists are announced. The ten best teams from the semifinals are invited to the final round. The host can invite some additional teams – usually to widen geographical representation or to present finals to schools not previously participated. There is a strict rule that at the final round, no more than two teams can represent the same school or out-of-school organization, like a computer club.

The number of teams and the best result for a particular round are shown in Table 1.

In total, 651 teams from 47 schools were in the semifinals, and 73 teams from 20 schools participated in Cēsis at the finals.

Table 1
Number of teams and schools represented at TCIM “Cēsis” and the best result

Year	Semifinals: teams/schools	The best result (max 1000 points)	Finals: teams/schools	The best result (max 1000 points)
2016	75/18	833	11/8	652.18
2017	100/27	860	10/9	715
2018	126/31	979	13/10	666
2019	118/25	814	13/10	607.44
2020	109/24	809	—	—
2022	40/15	988	11/7	863
2023	83/22	927	15/11	554.32

The schedule of finals is traditional – in the morning, all teams arrive in Cēsis, have a draw for a room where they work (each team in a separate room), and after the short opening ceremony, start working on tasks. For solving, 5.5 hours are given with a small break for lunch. After that jury proceeds with the evaluation of solutions, and at the closing ceremony, teams are awarded. All teams get diplomas – the first three place winners for medals, the following three get honorable mentions, and all others for participation. First-place winner also receives a trophy and is obliged to cut the winner’s cake in pieces to treat all participants and organizers.

It should be noted that, despite a high level of competition content, the atmosphere at the finals is friendly and informal. For example, traditionally printed task descriptions are not simply given to teams; they should solve a small task to deserve them. For example, in the final round of 2023, this task was to name any Latvian city having at least three letters shared with the name “CĒSIS.”

Traditionally, the strongest teams were from Riga State Gymnasium N1, which won 6 out of 7 finals, and only in 2019, the host team from Cēsis State Gymnasium won the competition.

Besides the contest part, serious organizational work is “behind the scenes.” Since TCIM is provided without financial support from the state, the successful provision of competition depends on the support of local enthusiasts from Cēsis State Gymnasium (with the direct support of directors Gunta Bērziņa and Ina Gaiķe), Cēsis municipality and sponsors. At the finals, there are “must have” things like a separate working room, one computer with Internet access for each team, and writing utensils and scratch paper. Traditionally, during finals, there is an excursion for teachers of participating teams and some activities for participants while submitted answers are evaluated. All participants of TCIM finals receive small memorabilia gifts at the end of the competition.

4. Tasks and Solutions

A lot about task types in TCIMs was written in previous papers (Opmanis, 2009; Opmanis, 2015).

There still are no widely known competitions in mathematics and informatics where unlimited usage of Internet resources is allowed. It should be emphasized that “Internet resources” assumes only access to materials and not the usage of the Internet as a communication environment to get help from outside. Allowance of Internet usage defines strict rules for task selection – tasks taken “as is” from public sources should be, in general, avoided (if it is not intended to search for this particular information). Published tasks can be given with some modifications or as subtasks. However, figures should not be copied but made from scratch to exclude direct matches with published sources. Even if the author of the task assumes that his task is original, it should be carefully checked that a solution can’t be obtained by a simple search.

Since task descriptions are prepared in Latvian, it simplifies the risk of finding a similar task compared to task descriptions in English. However, participant’s knowledge of English, the rapid evolution of translation and search systems, and artificial intelligence systems like ChatGPT make the preparation of task sets more and more challenging.

We will characterize some general aspects of tasks and expected answers:

Answers should be short in form – one or a few numbers, a short text string, some configuration of elements (like domino or pentamino pieces) or filled table (like SUDOKU). All possible correct answers should be found if it is not stated that it is enough to find any one of them. Traditionally it is not said whether there is one or several answers.

The number of tasks. In each round, ten tasks, each worth 100 points, are given.

Subtasks and grading schema. In the task set, tasks with “all or nothing” grading are rare – almost all tasks have 2–5 subtasks with partial scores. Different amounts of points are given for solved subtasks in various tasks, and this results in an excellent distribution of total points without (in finals) or rare (in semifinals) ties and, as a consequence, no problems with the determination of the winner or deciding the best teams.

Description of tasks. TCIM has much more freedom in task themes and task description formats than classic olympiads and contests like Bebras, where there is no space to describe unknown concepts or complicated rules. In Bebras, there is also time pressure since contestants should be able to comprehend and solve each task in 2–3 minutes.

TCIM has no formal limits on a description length or how standard language should be used. The main principle is that task descriptions should be clear and unambiguous. However, in the history of TCIM, there has been a case when due to text modification, the initial content of the task was slightly changed, leading to more than one correct answer. While the length of task descriptions is not limited, the usual length of a printed task set is 3 to 5 pages.

We feel free to use not-so-formal language. For example, in the task “Equivalent” (see below), after the observation that several jury members missed one particular clue, the last sentence was added.

While sometimes it is hard to label some tasks properly, the usual **task types** used are the following:

4.1. Data Processing/Data Mining Tasks

These tasks are traditional and were presented at all TCIM rounds. In these tasks file with some data or a link to the online data source is given, and teams are asked to explore these data and get answers to several questions. The intended tool is a spreadsheet, while it can be some database managing system or programming language. For example, at the finals of 2023, historical daily weather data for four years in five Latvian cities were obtained from the site “Visual Crossing Weather” (Visual Crossing), and one of the questions offered was: “In how many days there are no two cities with the same description of weather?”

It should be noted that since 2018 no team has gotten a full score in data processing tasks at TCIM finals.

4.2. Word Problems

Good tasks can be found in old journals and books. Since the Latvian National Digital Library (Periodika) contains scanned old periodicals, it is easy to find word problems even from the 19th century. One of such tasks from the “Rota” journal (issue 7, 1886) in its original form (including Gothic writing and rules to denote diacritical signs characteristic for this period) was given in the semifinal of 2017 (see Fig. 1), and 67% of teams solved this task.

A similar approach was used in the semifinals of 2023, where an even older task from the same journal (issue 1, 1884) in its original form was given. However, the success rate was lower this time – only 16,7% of teams solved this task. Of course, in these cases, the historical representation of the task is also essential.

4.3. Geometry

During the task preparation, several possible side effects should be considered. As a rule, there should be no answers that are easy to guess in geometrical tasks – like an angle of 30°, 45°, or 60°. From the experience of the last Latvian Math Olympiad (LMO), there may be an erroneous assumption that the answer should always be an integer. For example, if it is given that area of a rectangle is 20, then possible options for the shortest

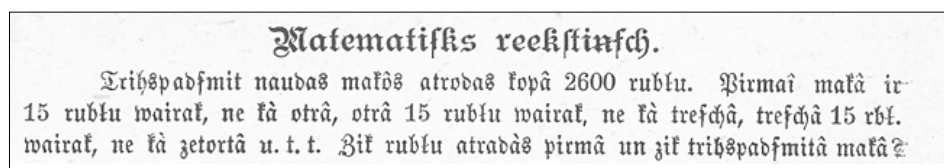


Fig. 1. The task from the journal “Rota.”

side can be only 1, 2, and 4. Therefore at TCIM, in geometrical tasks, almost all answers are real numbers, not integers, and an accuracy of 10^{-6} is required to get a full score. However, even this cannot help if the participant is familiar with software like GeoGebra in which you, in particular tasks, can get the proper answer just by the correct drawing.

Example of a task in geometry (semifinals of 2023):

In the convex quadrilateral $ABCD$, $\angle ABD = \angle ACD = 90^\circ$. Lengths of three edges are known: $AB = 487$ cm, $BC = 283$ cm, and $AD = 2022$ cm. Find the length of the edge CD in centimeters!

4.4. Combinatorics

Allowing the usage of Internet resources makes it more and more challenging to create tasks in combinatorics. For example, suppose it is asked to find several different combinations for particular parameter values. In that case, a standard way of solving may be to find “by hand” the answers for some small parameter values and use them to search for an appropriate integer sequence in the famous Online Encyclopaedia of Integer Sequences (OEIS®). If there are enough correct first members and the contestant can understand the mathematical language of how the particular sequence is described, the problem can likely be solved (at least partially). The task for problem setters now includes mandatory checking of OEIS® sources and (if possible) asking for solutions for values above those given in OEIS® or linked resources. It should be noted that finding an appropriate sequence, in general, is welcome, especially if a problem is formulated in a not straightforward form where the skill to understand that this is the correct sequence is deciding.

The task authors can be even more proud if the task has no corresponding sequence in OEIS® at the time when this task is given in a competition. One such task, “Guards of a castle,” was presented at the finals of 2023:

Four walls of the castle are defended by guards who may be positioned only in eight places – in towers at the corners or in the middle of each wall. Guards in the towers defend two appendant walls while guards in the middle of the wall guard only this wall. Guards should be distributed so that the same number of guards defends all four walls. If one distribution can be obtained from another by reflection and/or rotation, they are counted as one. For example, if there are six guards, there are five different distributions shown in Fig. 2.

What is the number of different distributions for a) 9, b) 30, c) 91, d) 2023 guards?

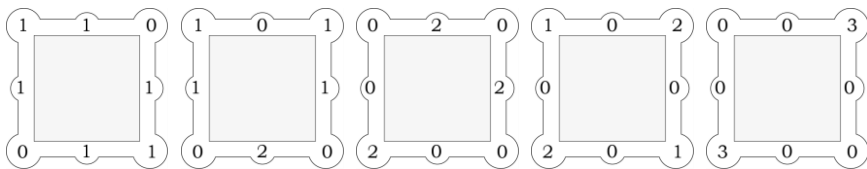


Fig. 2. Different distributions of six guards.

This task is interesting because you should write an effective computer program to solve subtasks b) to d) and be ready to wait for the computations to get the answer in subtask d). At the competition, only one team solved three first subtasks, but no one succeeded with the last one.

The task was inspired by tasks “The Courageous Garrison” and “Daylight lamps” (Kordemsky, 1972) where (in terms of our task) for the changing number of guards, it was asked to find a valid configuration for a particular constant number of guards defending each wall.

4.5. Number Theory

Immediate computer usage is not always the best option – from the viewpoint of time invested a “pure” mathematical approach can be more successful. At the finals of 2023, the task “Two last digits” was given:

Aivars investigates two increasing sequences of integers. In the sequence $\{p_i\} = \{2, 3, 5, 7, \dots\}$, he consecutively writes all prime numbers, and in the sequence $\{q_i\} = \{499999999, 589999999, 598999999, 599899999, 599989999, \dots\}$ – all integers with the sum of digits 76. After some time, Aivars notices a pair of consecutive numbers $(p_k = q_m, p_{k+1} = q_{m+1})$ shared between the sequences, and the difference between these numbers is 72.

What are the last two digits of p_k ?

One obvious solution is to write a computer program that simulates the two sequences in the description until the matching pair is found. However, finding the first matching pair will take quite a lot of computational time (on the only available computer!). Therefore, the best way is to get the only possible option by excluding all others and giving this as the answer.

Most probably, such a task will not be given at the pure mathematical competition because there it would not be enough to find the only possible candidate for the last two digits by excluding unacceptable options – you need to show also that such a number exists. To avoid the necessity to find the number itself, the task was reformulated from strict mathematical language to a story about some imaginary mathematician Aivars who found such a pair of numbers giving an indirect clue that there is a solution (maybe more than one).

4.6. Dominoes

Tasks about dominoes are traditionally used at almost all TCIM rounds. Sets of dominoes with various numbers of pieces appear in tasks alone or in combination with other classic games or puzzles (like SUDOKU at the finals of 2022 and chess at the finals of 2023).

Despite previous remarks about the need to avoid already published tasks, puzzle #492, “The domino column” from the book (Dudeney, 1976), was successfully used in the semifinals of 2020. While Dudeney’s book gives just one solution, it is easy to find general rules to obtain more solutions.

7	6	5	8	3	9	2		4	7	6	5	8	3	9	2	1	4
3		8	6	1	4	5	9	7	3	2	8	6	1	4	5	9	7
9	1	4	5	7	2		6	8	9	1	4	5	7	2	3	6	8
1	7	6	9	5	3		8	2	1	7	6	9	5	3	4	8	2
4	9	3	7	2	8	1	5	6	4	9	3	7	2	8	1	5	6
8		2	1	4		9		3	8	5	2	1	4	6	9	7	3
5	4	7	3		1	6	2	9	5	4	7	3	8	1	6	2	9
2	8		4	6	5	7	3		2	8	9	4	6	5	7	3	1
6	3	1		9	7	8	4	5	6	3	1	2	9	7	8	4	5

Fig. 3. Example of a valid LUDOKU puzzle and its solution.

4.7. SUDOKU

SUDOKU is a well-known logical puzzle, and nine tasks with its variations have been used at TCIM. One of its variations was the task “LUDOKU” (finals of 2016):

*Little Ludis invented a new logical puzzle based on SUDOKU and named it LUDOKU. LUDOKU is a traditional SUDOKU puzzle with the additional constraint that **the only solution** can be obtained by filling empty cells consecutively with numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, ... in the rows from top-down and each row from left to right. One valid LUDOKU puzzle and its solution are shown in Fig. 3.*

Create a LUDOKU puzzle with as many empty cells as possible! Be aware that the puzzle may have other solutions besides those obtained by LUDOKU rules!

4.8. Other Logical Puzzles

Variations of well-known puzzles besides SUDOKU are also given at TCIM rounds regularly.

For example, the previously mentioned task “Equivalent” (semifinals of 2023):

Fill squares in Fig. 4 with numbers from 1 to 6 so that:

- *In each row, all numbers are distinct.*
- *In each column, all numbers are distinct.*
- *If a line segment connects “diagonal neighbors” of two rows, then numbers in these squares are the same. It is known that **all** equivalent diagonal neighbors are marked.*

Five squares are already filled.

In the task description, there is one essential requirement that is missed by many readers (who are then surprised that they can't solve the task).

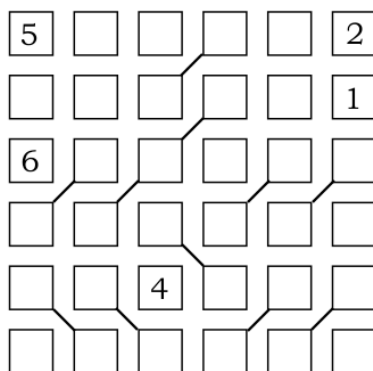


Fig. 4. Task “Equivalent”.



Fig. 5. Photo from the task “The Sorting Hat”.

4.9. Non-traditional Tasks

It should be noted that contestants enjoy non-traditional tasks very much.

At least twice, “think out of the box” tasks were given in the finals: to measure the precise volume of a trash bin (finals of 2018) and to estimate the length of yarn in a crocheted object (finals of 2019, see Fig. 5).

4.10. Search for Information

Especially popular (in the sense that all teams tried to solve them and got at least some partial scores) have been tasks connected to Internet searches. A current example of such a task is “Traffic accident” (finals of 2023), where a fragment from a photo (see Fig. 6)



Fig. 6. Photo from the task “Traffic accident”.

from a traffic accident was given, and it was asked when and where (country, precise coordinates) this accident took place.

It was assumed that contestants would find the proper place using “Google Maps” and a description of an accident in the local newspaper where the original photo was published. Even if contestants succeed and find the correct issue of the newspaper, they can easily lose points by missing the word “yesterday” in the accident description.

4.11. Tasks with the Unknown Best Answer

Strictly speaking, in all “classic” Math Olympiads and similar competitions, participants are solving tasks already solved previously by someone. At TCIM, tasks with the unknown best answer are not rare. For grading such tasks, formulae gave maximum points if the answer is best known at the moment of grading and proportionally fewer points if the answer does not reach this value.

One such task, “Felicitous pyramid,” was given at the finals of 2023:

A pyramid of integer numbers is built in the following way: At the beginning, N (where $N > 1$) circles in a row are drawn, and in each of them, some distinct integer is written. Then above the first row, there the next row of circles is drawn having one circle less and so that each circle is located in-between and above two circles of the previous row:



In the circle at the top (a) is written the absolute value of the difference between the two integers in circles located immediately below it ($|b - c|$). So rows are added and filled one by one until there is just one circle in the current row – the top of the pyramid is reached.

Let's say that a pyramid is felicitous if all numbers in it are distinct.

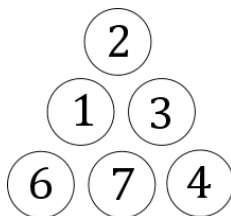


Fig. 7. Example of the felicitous pyramid for $N = 3$.

An example of a felicitous pyramid for $N = 3$ is shown in Fig. 7, and its largest integer is 7.

Your task is to create a felicitous pyramid with the largest integer as small as possible for a) $N = 6$, b) $N = 10$, c) $N = 15$.

If you know or find out during competition that such pyramids are called “anti-Pascal triangles” or “subtractive triangles,” then you can find (OEIS®, Cazor (2022)) the best answers for the first two subtasks. Unfortunately (for the participants), the answer for the last subtask still can’t be found there, and you should find some in any other way.

4.12. Just Try!

There are a lot of tasks where the best option is to take paper and pencil and start thinking about possible solutions.

One such task is: “Equivalent parts” (finals of 2023):

Each of the given five figures (see Fig. 8) containing 36 unit squares should be cut by lines into as few equivalent parts as possible (the same number of squares and identical or symmetrical form) so that these parts can be put together to form a 6×6 square. In the square, parts obtained may be rotated and/or flipped.

Each subtask should be solved separately.

Grading: 20 points for the subtask if the number of parts is the least known and 4 points if it is the next possible (under the given conditions, only possible numbers of parts in increasing order are 2-3-4-6-9-12-18-36). Solutions where there will be different parts will receive 0 points.

It can be added that no team got a full score on this task at the competition – the best result was 60 points.

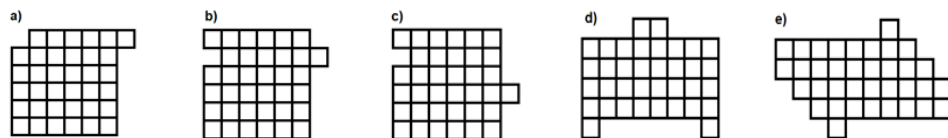


Fig. 8. Task “Equivalent parts” – given figures.

5. Evaluation and Grading

Evaluation of solutions at TCIM is challenging in two different ways: at the semifinals, there are many solutions, while at finals, solutions should be evaluated and graded in a short time interval (less than two hours). Without proper tools, it may lead to a heavy routine job at the semifinals or working in stressful conditions at finals. In both situations, there is a risk of making mistakes which can be crucial, especially in the final round. From the viewpoint of time invested, it was advantageous to implement an automatic evaluation and grading system for all competition rounds.

Diāna Siliņa suggested the usage of MS Excel as an evaluation and grading tool. She had previously successfully used the same approach for evaluating tests and homework in the gymnasium.

MS Excel and VBA spreadsheet applications were used for checking and evaluating answers at all TCIM rounds from the first of 2016 till the semifinals of 2023.

Automatic evaluation can have different implementations. If it is expected that everyone will be allowed to check their work individually, then it would be natural that the task, answer, test, and result are in one file. However, including confidential information in the file offered to the participants in graded tests and competitions is not wise.

Theoretically, part of the information can be hidden and the file protected with a password – but the risk remains that the password can be guessed. It is safer to put only the information the participants can see in the file intended for them (*answer file*). However, you still need a file that contains all the “magic” – the correct answers, the prepared table with empty cells for filling by the participants’ answers, formulas, condition formatting, and the START button (starts the execution of the VBA code. VBA is always used to record the results of the tasks in the summary table) – we will call this file the *evaluation file*.

The overall evaluation and grading process contains the following steps:

- 1) Open the evaluation file.
 - 2) Open all submitted answer files and click the START button in the evaluation file.
- After each submission is evaluated, a corresponding notification allows you to follow the progress. Creating backup copies of submissions and the result table would be advisable for security purposes.

The “evaluation difficulty” of the task has almost nothing in common with its “scientific difficulty.”

By classifying the tasks of previous years and collecting data in a table (see Table 2), a subjective assessment of the difficulty of implementing evaluation and grading has been given, which includes the time and knowledge necessary for the implementation of the evaluation.

Some comments about the difficulty levels in the table:

1. The fastest and easiest way to implement an automated evaluation is for tasks where the answer is unequivocal, entered in one cell of the spreadsheet, and only wholly correct answers are accepted by assigning 100 points.

Table 2
Estimation of the automated evaluation difficulty

Difficulty (1 – easy)	Task types	Spreadsheet functions (highest levels include lowers)	VBA	Time	Grading
1	Cryptarithms, geometry	SUM, AVERAGE, MIN, - MAX, IF, ROUND		+	Correct/incorrect – 100/0
2	Equations, cryptarithms, data analysis, searching for information	OR, AND, COUNTIF, - SUMPRODUCT		++	Contains subtasks
3	Geometry, data analysis	ABS, LOOKUP, GCD	-	+	The accuracy of the answer is rated.
4	Non-standard tasks		+	++	Answers may be in arbitrary order.
5		Supporting sheets	++	+++	

2. Little more time-consuming is the implementation of evaluation if the answer is unambiguous, but the task has partial scores, and/or the total number of points depends on the number of correctly solved sub-tasks.
3. The answer's accuracy is also considered during the evaluation.
4. Answers may sometimes be allowed to be entered in arbitrary order (it's easier not to allow this), so you need to sort them before comparing them with the set of correct answers. Participants may also find just some of the answers, so it may be necessary to review the textual information given by participants.
5. As it is usually more common in MS Excel to work with built-in functions than using VBA, in some cases, it is more convenient to copy answers from participant files to additional tables, which are located in the worksheet created for a specific task, and perform calculations/checks afterward with standard MS Excel tools. Then the results obtained from this worksheet can be copied into the results table using VBA.

Of course, not all tasks are suitable for a fully automated evaluation and grading. For example, in domino tasks, it is too hard to implement checking that the complete set of domino pieces is correctly used. At the same time, in such tasks, checking can be done at least partially (like checking that sums of points in domino halves fulfill some requirement).

Likewise, there is checking of the uniqueness of the solution in the SUDOKU tasks in VBA – such checkers can be found on the Internet, and their integration into the evaluation environment would be technically complicated.

Also, answers with various figures whose borders may change are not automatically evaluated.

The answer file should be carefully prepared, namely:

- Warn in the file that there will be automated evaluation, so the required input format should be strictly followed.
- Control/restrict data entry as much as possible.
- Strictly define the conditions for entering the answers, for example, forcing to enter the answers in alphabetical or ascending order.

- It is recommended to use the options offered by the spreadsheet application – drop-down lists, control of data to be entered (Data Validation), while at the same time providing an option to input large numbers or numbers with high accuracy as text.
- To reduce the possibility of entering information in an incorrect place, protect the worksheets (without a password, if an unexpected situation occurs that participants must be allowed to correct).
- Even a hidden type and password-protected workbook offered to participants should not contain correct answers or other confidential information.

Summary of the experience obtained from the preparation of the *evaluation file*:

- In all self-checking tasks, VBA is used to read information from answer files, process and compile it.
- If the answer to the task is short enough to be easily reviewed in the width of one screen, the information should be copied from the answer files to the evaluation file with the help of VBA – this helps to control the correctness of the evaluation, as well as quickly review all answers.
- If you can't check submitted answers automatically, a summary is helpful to review and evaluate all teams' responses manually.
- Use MS Excel's standard options where possible.
- If the number of points assigned depends on the number of correct answers, it is advantageous to use the LOOKUP function.
- Use conditional formatting for visual checking of evaluation correctness.

6. Further Development of the Evaluation System

The existing approach was successfully used till the semifinals of 2023 when Pēteris Pakalns suggested using a new web-based evaluation system, “UIM,” made by himself, at the finals of 2023.

This system was used without any problems, and imposing was its strength in tasks where an Excel-based system could not have been used, like in the already mentioned task, “Equivalent parts,” in which participants were able to define their configuration of a part on-screen and move, flip, and rotate to be sure that their solution works.

A screenshot from the “UIM” system submission page for the previously mentioned task “Equivalent parts” is given in Fig. 9.

The UIM system is currently undergoing active development, and the platform is accessible online (UIM). Within the platform, all tasks of the TCIM 2023 finals are available for public solving in the Latvian language. To view tasks in other languages, please utilize the translation options provided by your web browser.

The UIM platform was developed to evaluate solutions more promptly and identify potential problems during the contest. After the contest, it also gives participants a summary of testing results without a score and allows for timely appeals before the award ceremony and results announcement.

In the UIM system, all submissions are evaluated immediately after submission using a program written in the Rust programming language. The program is fed with the

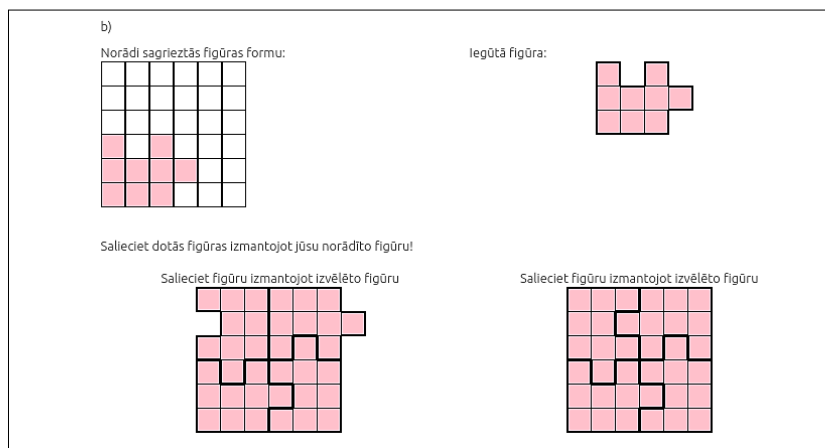


Fig. 9. Task “Equivalent parts” – definition of a part and assembled initial figure and a square.

definition of the answer form, the jury solution, and the participant solution. These programs validate the jury solution and participant solution based on the same criteria. If all criteria are passed, the participant’s score is calculated by comparing the processed jury and the participant’s answers. In addition, the UIM system provides a Rust library for writing submission evaluation programs, which makes the implementation process very ergonomic.

Due to the limited usage of the UIM system thus far, we have decided to postpone its detailed description until it reaches a more mature state after several (hopefully!) successful future TCIMs.

Acknowledgments

The authors would like to thank “Tom Sawyer Software Latvia” for their support of the finals of TCIM 2023 and Ahto Truu for his valuable comments.

References

- Bebras. International Challenge on Informatics and Computational Thinking. <https://www.bebbras.org/>
- Cazor, D. (2022). Filling a subtractive triangle. Minimal solutions. oeis.org/A226239/a226239_1.pdf.
- ChatGPT. <https://chat.openai.com/>
- Dudeney, H.E. (1967). *536 Puzzles and Curious Problems*. Charles Scribner’s sons. ISBN 0-684-71755-7, 198.
- GeoGebra. Tools for Teaching and Learning Math. <https://www.geogebra.org/>
- Kordemsky, B.A. (1972). *The Moscow Puzzles*. Charles Scribner’s sons. ISBN 0-684-14870-6, 37-38.
- LMO (n.d.). *Latvian Olympiad in Mathematics*. <https://www.nms.lv/olimpiades/valsts-olimpiade/>
- OEIS® (n.d.). *The Online Encyclopedia of Integer Sequences®*. <https://oeis.org>

- Opmanis, M. (2009). Team Competition in Mathematics and Informatics “Ugāle” – Finding New Task Types. *Olympiads in Informatics*, 3, 80–100.
- Opmanis, M. (2015). Math Contests: Solutions without Solving. *Olympiads in Informatics*, 9, 147–161.
- Periodika. Latvian National Digital Library, <http://www.periodika.lv/>
- UIM (n.d.). Automated competition platform UIM. <https://uim.aps.lv/>
- Visual Crossing. Global Forecast & History Data. <https://www.visualcrossing.com/weather-data>



M. Opmanis is a researcher at the Institute of Mathematics and Computer Science of the University of Latvia. He is head of the jury of Latvian OI and TCIM, and a lot of times, was a leader of the Latvian team at IOI and Baltic OI. He is the author of tasks for various contests in informatics and mathematics.



D. Siliņa is an informatics and programming teacher (since 1995) at Cēsis State Gymnasium and one of the key organizers of TCIM. She was the leader of the Latvian team at Baltic OI in 2012 and 2020 and the main organizer of Latvian OI in 2011 hosted by Cēsis State Gymnasium.



S. Siliņa is a leading programming technician at the Institute of Mathematics and Computer Science of the University of Latvia. She graduated from Cēsis State Gymnasium and is the winner of TCIM 2019. Sandra works on the TCIM and Latvian OI jury, and she was the deputy leader of the Latvian team at European Girls OI in 2022.



P. Pakalns graduated from Cēsis State Gymnasium and won TCIM “Ugāle” in 2014. Now he works on the TCIM and Latvian OI jury and was the leader of the Latvian team at Baltic OI. He is the developer of an automated system for evaluating student programming assignments, used at the Faculty of Computer Science at the University of Latvia, evaluating over 20,000 student submissions. He is the author of the new automated competition platform UIM.

REVIEWS, COMMENTS

Guide to Teaching Data Science: An Interdisciplinary Approach

Orit HAZZAN, Koby MIKE

Faculty of Education in Science and Technology

Technion – Israel Institute of Technology

e-mail: oritha@technion.ac.il, mike@technion.ac.il, kobymike@gmail.com

Introduction

Data science is a new discipline of research that is gaining growing interest in both industry and academia. Data science is converging knowledge, skills and values from computer science, statistics, and various applications domains such as social science, digital humanities, life science and more (see Fig. 1).

As a result of the growing interest in data science, demand is increasing for data science programs for a variety of learners from a variety of disciplines (data science, computer science, statistics, engineering, life science, social science and humanities) and a variety of levels (from school children to academia and industry). While significant efforts are being invested in the development of data science curricula, or in other words, in *what to teach*, only sporadic discussions focus today on the data science pedagogy, that is, on *how to teach*. This is the focus of our recently (March 2023) published book by Springer: *Guide to Teaching Data Science: An Interdisciplinary Approach*, described in this paper.

The guide can be used by all educators in all educational environments and settings: in formal education (from elementary schools through high schools to academia) and informal education, in industry, and in non-profit governmental and third sector organizations. Specifically, the guide can be used as a textbook for Methods of Teaching Data

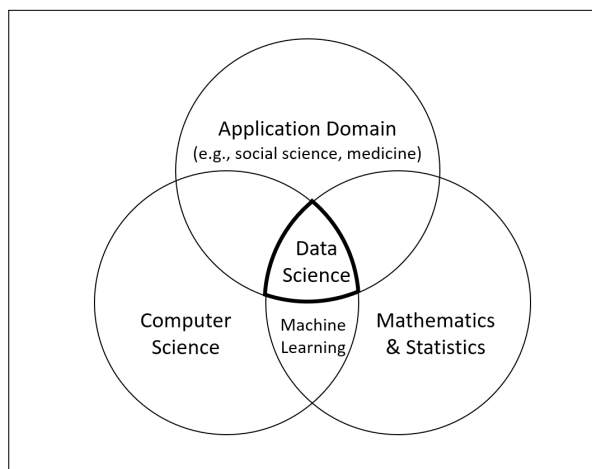


Fig. 1. The authors' version of the data science Venn diagram, as inspired by Conway (2013).

Science courses, in which prospective and in-service teachers learn the pedagogy of data science, which is currently emerging in parallel to the development of the discipline of data science.

To benefit all of its potential user populations, the guide is organized in a way that enables immediate application of its main ideas. This goal is achieved by presenting the rationale behind the inclusion of each topic presented in this guide, its background, development, and importance in the context of data science and data science education, as well as the details of the actual teaching process (including over 200 exercises, worksheets, topics for discussions, and more).

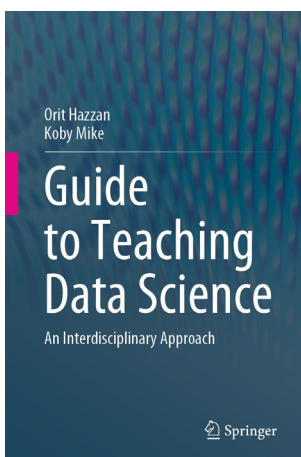


Fig. 2. Guide to Teaching Data Science: An Interdisciplinary Approach – Front Cover.

Description of the Guide parts and chapters

The guide is divided into five parts.

- **Part A – Overview of Data Science and Data Science Education.** In this part, we discuss what data science and data science education are and review the current state of data science education, including curricula and pedagogy.
- **Part B – Opportunities and Challenges of Data Science Education.** This part of the guide elaborates on the educational challenges of data science, from a variety of perspectives (including learners, teachers, and policy makers among other), addressing also the multi-faceted and interdisciplinary nature of data science.
- **Part C – Teaching Professional Aspects of Data Science.** In this part, we examine several topics related to the human aspects of data science, such as data science skills and social issues, in general, and ethics, in particular. We highlight the message that data science skills and other non-technical issues should be given special attention in any data science program regardless of its framework or level.
- **Part D – Machine Learning Education.** We dedicate one part of this guide to machine learning education for two main reasons. First, machine learning is one of the central steps in the data science workflow and an important and central emerging approach for data modeling. Second, machine learning is heavily based on mathematics and computer science content, and therefore poses unique pedagogical challenges that we address in this part of the guide. Specifically, we discuss teaching machine learning algorithms using a white box approach, teaching core concepts of machine learning algorithms that are commonly taught in introductory data science courses, and specific teaching methods suitable for teaching machine learning.
- **Part E – Frameworks for Teaching Data Science.** This part of the guide presents several frameworks for teaching data science to professionals whose core activities are management, education, or research, and who need data science knowledge to foster their professional development and improve their professionalism.

In what follows we present the book chapters along with 1 or 2 illustrative exercises from each chapter. From the variety of over 200 exercises included in the guide, we chose to present in this paper exercises that fit the readership of the *Olympiads in Informatics* journal.

Chapter 1. Introduction – What Is This Guide About?

In the introduction, we present the motivation for writing this guide, followed by the pedagogical principles we applied in it, its structure and how it can be used by educators who teach data science in different educational frameworks. We also present sev-

eral main kinds of learning environments that are appropriate for teaching and learning data science: Textual programming environments for data science¹ such as, Jupyter Notebook¹ and Google Colab² and visual programming environments for data science, such as, Orange Data Mining³, KNIME⁴, and Weka⁵.

Illustrative exercise

Reflection

Reflect on what you have read so far:

- (a) What pedagogical ideas were introduced in this chapter?
- (b) Can you speculate how *you* will use this guide when you teach data science (now or in the future)?

Part A – Overview of Data Science and Data Science Education

This part includes the following chapters:

Chapter 2: What is Data Science?

Chapter 3: Data Science Thinking

Chapter 4: The Birth of a New Discipline: Data Science Education

Chapter 2. What is Data Science?

Data science integrates knowledge and skills from several disciplines, namely computer science, mathematics, statistics, and an application domain. One way to present such a relationship is using a Venn diagram, which is a diagram that shows logical relationships between different sets (See Fig. 1).

Although many attempts have been made to define data science, such a definition has not yet been reached. One reason for the difficulty to reach a single, consensus definition for data science is its multifaceted nature: it can be described as a science, as a research method, as a discipline, as a workflow, or as a profession. One single definition just cannot capture this diverse essence of data science. In this chapter, we first review the background for the development of data science. Then we present data science from several perspectives: data science as a science, data science as a research method, data science as a discipline, data science as a workflow, and data science as a profession (Mike and Hazzan, 2023). We conclude by highlighting three main characteristics of data science: interdisciplinarity, learner diversity, and its research-oriented nature.

¹ <https://jupyter.org/>

² <https://colab.research.google.com/>

³ <https://orange.biolab.si/>

⁴ <https://www.knime.com/>

⁵ <https://www.cs.waikato.ac.nz/ml/index.html>

Illustrative exercise**Pedagogical implications of the multi-faceted analysis of data science**

What pedagogical implications can you draw from the analysis of data science as a science? as a research method? as a discipline? as a workflow? as a profession? What mutual relationships exist between these implications?

Chapter 3. Data Science Thinking

This chapter highlights the cognitive aspect of data science. It presents a variety of modes of thinking, which are associated with the different components of data science, and describes the contribution of each one to data thinking – the mode of thinking required of data scientists (not only professional ones). Indeed, data science thinking integrates the thinking modes associated with the various disciplines that make up data science. Specifically, computer science contributes computational thinking, statistics contributes statistical thinking, mathematics adds different ways in which data science concepts can be conceived, and each application domain brings with it its thinking skills, core principles, and ethical considerations. Finally, we present data thinking. The definition of data science inspires the message that processes of solving real-life problems using data science methods should not be based only on algorithms and data, but also on the application domain knowledge (Mike *et al.*, 2022).

Illustrative exercise**Additional modes of thinking required for data science**

Different publications on data science skills mention different thinking skills as being required in order to deal meaningfully with data science. These include, among others, analytical thinking, critical thinking, and data literacy.

Explore these thinking skills (and others you may find) as well as the interconnections between them and the various thinking skills presented in this chapter.

Chapter 4. The Birth of a New Discipline: Data Science Education

Data science is a young field of research and its associated educational knowledge – data science education – is even younger. As of the time of writing this book, data science education has not yet gained recognition as a distinct field and is mainly discussed in the context of the education of the disciplines that make up data science, i.e., computer science education, statistics education, mathematics education, and the educational fields of the applications domains, such as medical education, business analytics education, or learning analytics. In this chapter, we present the story of the birth of the field of data science education by describing its short history. We focus on the main

efforts invested in the design of an undergraduate data science curriculum, and on the main initiatives aimed at tailoring a data science curriculum for school pupils. We also suggest several meta-analysis exercises that examine these efforts.

Illustrative exercise

Didactic transposition in data science

Didactic Transposition is a concept that refers to the process of adopting knowledge used by practitioners for teaching purposes (Chevallard, 1989). The term was first coined in the context of mathematics education, in which it refers to the process by which formal mathematics is adapted to fit school teaching and learning. For example, the introduction of a proof using two columns, one for the statement and the other for reasoning, represents “a didactic transposition from abstract knowledge about mathematical proofs” (Kang and Kilpatrick, 1992, p. 3).

- (a) Suggest several examples of didactic transpositions of formal mathematics to school mathematics. Reflect: What guidelines did you follow?
- (b) Read the paper by Hazzan et al (2010) in which the authors demonstrate didactic transpositions of software development methods to educational frameworks. What are the paper’s main messages?
- (c) Suggest possible didactic transpositions of data science concepts from the academia to high school and elementary school.

Part B – Opportunities and Challenges of Data Science Education

This part includes the following chapters:

Chapter 5: Opportunities in Data Science Education

Chapter 6: The Interdisciplinarity Challenge

Chapter 7: The Variety of Data Science Learners

Chapter 8: Data Science as a Research Method

Chapter 9: The Pedagogical Chasm in Data Science Education

Chapter 5. Opportunities in Data Science Education

Data science education opens up multiple new educational opportunities. In this chapter, we elaborate on six such opportunities: teaching STEM in a real-world context, teaching STEM with real-world data, bridging gender gaps in STEM education, teaching 21st century skills, interdisciplinary pedagogy, and professional development for teachers. We conclude with an interdisciplinary perspective on the opportunities of data science education.

Illustrative exercise**Teaching the STEM subjects in a real-world context**

Review the different topics you teach as part of your disciplinary teaching.

Select one of these topics and determine whether or not you currently teach it in the context of real world. If you teach it in a real-world context, choose another topic. Repeat this process until you find a topic that you do not teach in a real-world context.

Describe how you currently teach this topic and design a new teaching process for it, in a real-world context. Compare the two teaching processes. What are your conclusions? Suggest some general guidelines for teaching different subject matters in a real-world context.

Chapter 6. The Interdisciplinarity Challenge

In this chapter, we elaborate on the challenges posed by the interdisciplinary structure of data science. First, we describe the unique and complex interdisciplinary structure of data science. Then, we present the challenge of balancing computer science and statistics in data science education, and the challenge of actually integrating the application domain knowledge into data science study programs, courses, and student projects.

Illustrative exercise**Data science PCK**

Imagine you are a data science teacher. Describe your teaching environment, according to your choice: characterize the students, define the study program, describe the physical learning environment, etc.

What pedagogical-content knowledge PCK (Shulman, 1986) would you need in order to teach this class? Describe scenarios in which this PCK might be expressed in your teaching.

Chapter 7. The Variety of Data Science Learners

Since data science is considered to be an important 21st century skill, it should be acquired by everyone - children as well as adults – on a suitable level, to a suitable breadth, and to a suitable depth. And so, after reviewing the importance of data science knowledge for everyone, this chapter reviews the teaching of data science to different populations: K-12 pupils in general and high school computer science pupils in particular, undergraduate students, graduate students, researchers, data science educators, practitioners in the industry, policy makers, users, and the general public. For each population, we discuss the main purpose of teaching it data science, main concepts that the said population should learn and (in some cases) learning environments and exercises that fit it.

Illustrative exercises

The AI + Ethics Curriculum for Middle School initiative

The “AI + Ethics Curriculum for Middle School” initiative, presented at <https://www.media.mit.edu/projects/ai-ethics-for-middle-school/overview/>, focuses on artificial intelligence. It seeks to develop an open-source curriculum for middle school students that is made up of a series of lessons and activities.

Explore the activities proposed by this initiative.

In your opinion, what were the pedagogical guidelines applied in the development of these activities? Can these guidelines be applied in the development of learning material that focuses on other data science topics?

Chapter 8. Data Science as a Research Method

In this chapter, we focus on the challenges that emerge from the fact that data science is also a research method. First, we describe the essence of the research process that data science inspires. Then, we present examples of cognitive, organizational, and technological skills which are important for coping with the challenge of data science as a research method, and highlight pedagogical methods for coping with it. In the conclusion of this chapter, we review, from an interdisciplinary perspective, the skills required to perform data science research.

Illustrative exercise

The challenge of leaning the application domain

As can be seen in Chapter 7, a variety of populations nowadays study data science. How can the challenge of familiarity with the application domain be overcome when a specific population studies data science research methods but lacks the required application domain knowledge?

Chapter 9. The Pedagogical Chasm in Data Science Education

As an interdisciplinary discipline, data science poses many challenges for teachers. This chapter presents the story of one of them, specifically of the adoption of a new data science curriculum developed in Israel for high school computer science pupils, by high school computer science teachers. We analyze the adoption process using the diffusion of innovation and the crossing the chasm theories. Accordingly, we first present the diffusion of innovation theory and the crossing the chasm theory. Then, we present the data science for high school curriculum case study. Data collected from teachers who learned to teach the program reveals that when a new curriculum is adopted, a *pedagogical chasm* might exist (i.e., a pedagogical challenge that reduces the motiva-

tion of most teachers to adopt the curriculum) that slows down the adoption process of the innovation. Finally, we discuss the implications of the pedagogical chasm for data science education.

Illustrative exercise

Reflection on your experience with the adoption of innovation

According to the Diffusion of Innovation theory (Rogers, 1962), innovations spread in society by flowing from one of the following five distinct groups of adopters to the next: Innovators, early adopters, early majority, late majority, laggards.

- (a) Explore the characteristics of each group.
- (b) Reflect on your personal experience with the adoption of innovations. What group did you belong to in each case? Was it the same group? Were they different groups? What can you learn about your personality as an adaptor of innovation?

Part C – Teaching Professional Aspects of Data Science

This part includes the following chapters:

Chapter 10: The Data Science Workflow

Chapter 11: Professional Skills and Soft Skills in Data Science

Chapter 12: Social and Ethical Issues of Data Science

Chapter 10. The Data Science Workflow

The examination of data science as a workflow is yet another facet of data science. In this chapter we elaborate on the data science workflow from an educational perspective. First, we present several approaches to the data science workflow, following which we elaborate on the pedagogical aspects of the different phases of the workflow: data collection, data preparation, exploratory data analysis, modeling, and communication and action. We conclude with an interdisciplinary perspective on the data science workflow.

Illustrative exercises

Data preparation

Search for a dataset in an application domain you are familiar with. Review the data. Can you find erroneous data? Can you find outliers?

Search for a dataset in an application domain you are not familiar with. Review the data. Can you find erroneous data? Can you find outliers?

- (a) Analyze differences (if such exist) between the results in the two cases: the familiar application domain and the unfamiliar application domain.

- (b) Reflect on how you performed this exercise: How did you look for erroneous data? How did you look for outliers? Did you use any resources in these processes? If you did, which resources? If not, why not? Can these processes be improved? If yes, how? If not, why?

Chapter 11. Professional Skills and Soft Skills in Data Science

Abstract In this chapter, we highlight skills that are required to deal with data science in a meaningful manner. The chapter describes two kinds of skills: professional skills and soft skills. Professional skills are specific skills that are needed in order to engage in data science, while soft skills are more general skills that acquire unique importance in the context of data science. In each section, we address both cognitive, organizational, and technological skills.

Illustrative exercises

Critical thinking

One of the most important cognitive skills required for dealing meaningfully with data science ideas is critical thinking. Propose a case study of a data science project in which decision-making processes, which were not accompanied with critical thinking processes, led to a chain of undesirable events.

Chapter 12. Social and Ethical Issues of Data Science

The teaching of social issues related to data science should be given special attention regardless of the framework or level at which data science is taught. This assertion is derived from the fact that data science (a) is relevant for many aspects of our lives (such as health, education, social life, and transportation); (b) can be applied in harmful ways (even without explicit intention); and (c) involves ethical considerations derived from the application domain. Of the many possible social topics whose teaching might have been discussed in this chapter, we focus on data science ethics. We also present teaching methods that are especially appropriate for the teaching of social issues of data science.

Illustrative exercise

Famous cases that illustrate the need for an ethical code for data science

- (a) Locate resources about the two cases mentioned above that illustrate the need for an ethical code for data science. List the ethical considerations involved in each case.
- (b) Find additional cases that illustrate the importance of an ethical code for data science.

Part D – Machine Learning Education

This part comprises the following chapters:

- Chapter 13: The Pedagogical Challenge of Machine Learning Education
- Chapter 14: Core Concepts of Machine Learning
- Chapter 15: Machine Learning Algorithms
- Chapter 16: Teaching Methods for Machine Learning

Chapter 13. The Pedagogical Challenge of Machine Learning Education

Machine learning (ML) is the essence of the modeling phase of the data science workflow. In this chapter, we focus on the pedagogical challenges of teaching ML to various populations. We first describe the terms *white box* and *black box* in the context of ML education. Next, we describe the pedagogical challenge with respect to different learner populations including data science major students as well as non-major students. Then, we present three framework remarks for teaching ML (regarding statistical thinking, interdisciplinary projects, and the application domain knowledge), which are important to be kept in mind in ML teaching processes. We conclude this chapter by highlighting the importance of ML education in the context of the application domain.

Illustrative exercise

The concepts of *explainability* and *interpretability*

Search the web and find 3-5 stories that exemplify the concepts of *explainability* and *interpretability*.

- (a) For each story, identify its main actors, the ML algorithm it refers to, the context in which these concepts are discussed, the end of the story, and what conclusions are drawn (if at all).
- (b) Add your conclusions from the examination of each story.
- (c) Formulate three guidelines for users of ML methods.
- (d) Reflect on what you have learned while working on this exercise. What would you do differently if you were asked to repeat it?
- (e) What conclusions can you draw for your own usage of ML results?

Chapter 14. Core Concepts of Machine Learning

In this chapter, we focus on the teaching of several core concepts that are common to many machine learning (ML) algorithms (such as hyper-parameter tuning) and, as such, are essential learning goals in themselves, regardless of the ML algorithms. Specifically, we discuss types of ML, ML parameters and hyperparameters, model training, validation,

and testing, ML performance indicators, bias and variance, model complexity, overfitting and underfitting, loss function optimization and the gradient descent algorithm, and regularization. We conclude this chapter by emphasizing what ML core concepts should be discussed in the context of the application domain.

Illustrative exercise

True or false

- (a) Define the concepts true-positive, true-negative, false-positive, and false-negative. Which of these concepts does the medical diagnosis problem use?
- (b) Select a problem from any domain of life, whose formulation includes these concepts. Formulate it in two ways: using frequencies and using probabilities (percentages).
- (c) If you are working in a team, the team can discuss the different problems, addressing questions such as: In what context is each formulation clearer? Was it easy to transition between the two formulations? Why?

Chapter 15. Machine Learning Algorithms

In this chapter, we describe the teaching of several machine learning (ML) algorithms that are commonly taught in introduction to ML courses, and analyze them from a pedagogical perspective. The algorithms we discuss are the K-nearest neighbors (KNN), decision trees, Perceptron, linear regression, logistic regression, and neural networks.

The exercises in this chapter are based on explorations whose length and depth are beyond the scope of this paper.

Chapter 16. Teaching Methods for Machine Learning

In this chapter, we review four teaching methods for machine learning: visualization, hands-on tasks, programming tasks, and project-based learning. When relevant, as part of the presentation of these pedagogical tools, we analyze them from the perspective of the process-object duality theory and the reduction of abstraction theory.

The exercises in this chapter are based on explorations whose length and depth are beyond the scope of this paper.

Part E – Frameworks for Teaching Data Science

This part includes the following chapters:

Chapter 17: Data Science for Managers and Policymakers

Chapter 18: Data Science Teacher Preparation: The “Method for Teaching Data Science” Course

Chapter 19: Data Science for Social Science and Digital Humanities Research

Chapter 20: Data Science for Research on Human Aspects of Science and Engineering

Chapter 17. Data Science for Managers and Policymakers

In this chapter, we describe a workshop for policy makers that focuses on the integration of data science into education systems for policy, governance, and operational purposes. The messages conveyed in this chapter can be applied in other systems and organizations in all sectors – governmental (the first sector), for-profit organizations (the second sector), and non-profit organizations (the third sector). We conclude with an interdisciplinary perspective on data science for managers and policymakers.

Illustrative exercise

Data culture

Explore the term *data culture*. Use at least three resources that address this concept.

- (a) Give five examples of organizations that promote a healthy data culture.
- (b) List five characteristics of organizations that promote a healthy data culture.
- (c) Describe five practices that employees in organizations that promote healthy data cultures should master.
- (d) Describe five scenarios involving managers in organizations that promote a healthy data culture that illustrate the relevance of data science for their decision-making processes.

For each scenario, specify the data science knowledge the managers should have and suggest how and what they can learn from this specific data science content.

- (e) Explore the concept of exponential organizations (Ismail, 2014). Exponential Organizations: Why new organizations are ten times better, faster, and cheaper than yours (and what to do about it), Diversion Books). How do exponential organizations promote data culture?

Chapter 18. Data Science Teacher Preparation: The “Method for Teaching Data Science” Course

In this chapter, we present a detailed description of the Method for Teaching Data Science (MTDS) course that we designed and taught to prospective computer science teachers at our institution, the Technion – Israel Institute of Technology. Since our goal in this chapter is to encourage the implementation and teaching of the MTDS course in different frameworks, we provide the readership with as many details as possible about the course, including the course environment, the course design, the learn-

ing targets and structure of the course, the grading policy and assignments, teaching principles we employed in the course, and a detailed description of two of the course lessons. Full, detailed descriptions of all 13 course lessons are available on our Data Science Education website⁶.

Illustrative exercise

Topics to be included in a Method of Teaching Data Science course

Before reading the description of the course lessons, suggest topics that you would include in a Methods of Teaching Data Science course.

Chapter 19. Data Science for Social Science and Digital Humanities Research

In this chapter and in Chapter 20, we describe two data science teaching frameworks for researchers: this chapter addresses researchers in social science and digital humanities; Chapter 20 addresses researchers in science and engineering. Following a discussion of the relevancy of data science for social science and digital humanities researchers, we describe a data science bootcamp designed for researchers in those areas. Then, we present the curriculum of a year-long specialization program in data science for graduate psychology students that was developed based on this bootcamp. Finally, we discuss the data science teaching frameworks for researchers in social science and digital humanities from motivational perspectives and conclude by illuminating the importance of an interdisciplinary approach in designing data science curricula for application domain specialists.

Illustrative exercise

Data science applications that require knowledge in social sciences and digital humanities

- (a) Search the web for data science applications whose development required knowledge in social sciences. What do these applications have in common? In what ways are they different?
- (b) Search the web for data science applications whose development required knowledge in digital humanities? What do these applications have in common? In what ways are they different?
- (c) Are there similarities between the applications that require knowledge in the social sciences and the applications that require knowledge in the digital humanities? If yes – what are the similarities? If not – explain the differences between these two.

⁶ <https://orithazzan.net.technion.ac.il/data-science-education/>

Chapter 20. Data Science for Research on Human Aspects of Science and Engineering

In this chapter and in Chapter 19, we describe two data science teaching frameworks for researchers: Chapter 19 addresses researchers in social science and digital humanities; this chapter addresses science and engineering researchers and discusses how to teach data science methods to science and engineering graduate students to assist them in conducting research on human aspects of science and engineering. In most cases, these target populations, unlike the community of social scientists (discussed in Chapter 19), have the required background in computer science, mathematics, and statistics, and need to be exposed to the human aspects of science and engineering which, in many cases, are not included in scientific and engineering study programs. We start with the presentation of possible human-related science and engineering topics for investigation. Then, we describe a workshop for science and engineering graduate students that can be facilitated in a hybrid format, combining synchronous (online or face to face) and asynchronous meetings. We conclude with an interdisciplinary perspective of data science for research on human aspects of science and engineering.

Illustrative exercise

Data-driven research

- (a) Suggest five research topics in *scientific* disciplines that may be initiated by data that is gathered incidentally.
- (b) Suggest five research topics in *engineering* disciplines that may be initiated by data that is gathered incidentally.
- (c) For each topic, suggest a human-related topic that you would find interesting to investigate.
- (d) Select two topics from the human-related scientific disciplines and two topics from the human-related engineering disciplines and describe how you would research them.

Epilogue

In the epilogue of the *Guide to Teaching Data Science: An Interdisciplinary Approach*, we view it from a holistic perspective, reflecting on its big ideas and their interconnections, highlighting the following facts:

- The guide is multifaceted and addresses teaching methods, skills, learners, perspectives, habits of mind, and data science topics – from programming and statistics, through problem solving processes to organizational skills.
- The guide reflects the richness of the discipline of data science, its relatedness to many aspects of our life, and its centrality and potential contribution to the education of future generations in the 21st century.

- The richness of the discipline of data science is reflected in the interconnections between the different chapters of the guide, as they are specified throughout the guide.

Illustrative exercise

Final reflection task

Reflect:

- What do you like about data science? What do you dislike about it?
- What do you like about data science education? What do you dislike about it?
- If you had to formulate one main idea of data science education, what would it be?
- What will *your* main new contribution to data science education be?

Conclusion

In this paper we present the content of the *Guide to Teaching Data Science: An Interdisciplinary Approach*. We hope that it reflects the richness of data science and of data science education as emerging disciplines. Supplementary pedagogical material is available in our website at <https://orithazzan.net.technion.ac.il/data-science-education/>. We will be happy to continue the dialogue with the readership of the *Olympiads in Informatics* journal. Specifically, we welcome questions and suggestions for pedagogical tools and teaching methods as well as suggestions for collaboration for the promotion of data science education in the interdisciplinary spirit suggested in the guide.

References

- Chevallard, Y. (1989). On didactic transposition theory: Some introductory notes. In: *Proceedings of the International Symposium on Selected Domains of Research and Development in Mathematics Education*. 51–62.
- Conway, D. (2013). The data science venn diagram. *Datist*.
<http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>
- Hazzan, O., Dubinsky, Y., Meerbaum-Salant, O. (2010). Didactic transposition in computer science education. *ACM Inroads*, 1(4), 33–37.
- Ismail, S. (2014). *Exponential Organizations: Why New Organizations are Ten Times Better, Faster, and Cheaper than Yours (and what to do about it)*. Diversion Books.
- Kang, W., Kilpatrick, J. (1992). Didactic transposition in mathematics textbooks. *For the Learning of Mathematics*, 12(1), 2–7.
- Mike, K., Hazzan, O. (2023). What is Data Science? *Communications of the ACM*, 66(2), 12–13.
- Mike, K., Ragonis, N., Rosenberg-Kima, R.B., Hazzan, O. (2022). Computational thinking in the era of data science. *Communications of the ACM*, 65(8), 31–33.
- Rogers, E.M. (1962). *Diffusion of Innovations*. Free Press.
- Shulman, L.S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, 15(2), 4–14.



O. Hazzan is a faculty member at the Technion's Department of Education in Science and Technology since October 2000. Her research focuses on computer science, software engineering and data science education. Within this framework she researches cognitive and social processes on the individual, the team and the organization levels, in all kinds of organizations. She has published about 130 papers in professional refereed journals and conference proceedings, and eight books. In 2007-2010 she chaired the High School Computer Science Curriculum Committee assigned by the Israeli Ministry of Education. In 2011-2015 Hazzan was the faculty Dean. From 2017 to 2019, Hazzan served the Technion Dean of Undergraduate Studies. Additional details can be found in her personal homepage: <https://orithazzan.net.technion.ac.il/>



K. Mike is a Ph.D. graduate from the Technion's Department of Education in Science and Technology under the supervision of Professor Orit Hazzan. He continued his a post-doc research on data science education at the Bar-Ilan University, and retains B.Sc. and an M.Sc. in Electrical Engineering from Tel Aviv University. After two decades of professional career in the Israeli hi-tech industry, he returned to academia for his doctoral studies on data science education. As part of his research, Koby developed and taught several data science programs for high school students, high school computer science teachers, and graduate students and researchers in social sciences and digital humanities.

About Journal and Instructions to Authors

OLYMPIADS IN INFORMATICS is a peer-reviewed scholarly journal that provides an international forum for presenting research and developments in the specific scope of teaching and learning informatics through olympiads and other competitions. The journal is focused on the research and practice of professionals who are working in the field of teaching informatics to talented student. OLYMPIADS IN INFORMATICS is published annually (in the summer).

The format for the journal follows the tracks:

- the primary section of the journal focuses on research
- the second report section is devoted to sharing experiences of countries in informatics olympiads
- the last smallest section presents books reviews or other information

The journal is closely connected to the scientific conference annually organized during the International Olympiad in Informatics (IOI).

Abstracting/Indexing

OLYMPIADS IN INFORMATICS is abstracted/indexed by:

- Cabell Publishing
- Central and Eastern European Online Library (CEEOL)
- EBSCO
- Educational Research Abstracts (ERA)
- ERIC
- InfoBase Index
- INSPEC
- SCOPUS – Elsevier Bibliographic Databases

Submission of Manuscripts

All research papers submitted for publication in this journal must contain original unpublished work and must not have been submitted for publication elsewhere. Any manuscript which does not conform to the requirements will be returned.

The journal language is English. No formal limit is placed on the length of a paper, but the editors may recommend the shortening of a long paper.

Each paper submitted for the journal should be prepared according to the following structure:

- concise and informative title
- full names and affiliations of all authors, including e-mail addresses

- informative abstract of 70–150 words
- list of relevant keywords
- full text of the paper
- list of references
- biographic information about the author(s) including photography

All illustrations should be numbered consecutively and supplied with captions. They must fit on a 124 × 194 mm sheet of paper, including the title.

The references cited in the text should be indicated in brackets:

- for one author – (Johnson, 1999)
- for two authors – (Johnson and Peterson, 2002)
- for three or more authors – (Johnson *et al.*, 2002)
- the page number can be indicated as (Hubwieser, 2001, p. 25)

The list of references should be presented at the end of the paper in alphabetic order. Papers by the same author(s) in the same year should be distinguished by the letters a, b, etc. Only Latin characters should be used in references.

Please adhere closely to the following format in the list of references:

For books:

Hubwieser, P. (2001). *Didaktik der Informatik*. Springer-Verlag, Berlin.

Schwartz, J.E., Beichner, R.J. (1999). *Essentials of Educational Technology*. Allyn and Bacon, Boston.

For contribution to collective works:

Batissta, M.T., Clements, D.H. (2000). Mathematics curriculum development as a scientific endeavor. In: Kelly, A.E., Lesh, R.A. (Eds.), *Handbook of Research Design in Mathematics and Science Education*. Lawrence Erlbaum Associates Pub., London, 737–760.

Plomp, T., Reinen, I.J. (1996). Computer literacy. In: Plomp, T., Ely, A.D. (Eds.), *International Encyclopedia for Educational Technology*. Pergamon Press, London, 626–630.

For journal papers:

McCormick, R. (1992). Curriculum development and new information technology. *Journal of Information Technology for Teacher Education*, 1(1), 23–49.

<http://rice.edn.deakin.edu.au/archives/JITTE/j113.htm>

Burton, B.A. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4, 3–14.

For documents on Internet:

IOI (2008). *International Olympiads in Informatics*

<http://www.IOInformatics.org/>

Hassinen, P., Elomaa, J., Ronkko, J., Halme, J., Hodju, P. (1999). *Neural Networks Tool – Nenet (Version 1.1)*.

<http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html>

Authors must submit electronic versions of manuscripts in PDF to the editors. The manuscripts should conform all the requirements above.

If a paper is accepted for publication, the authors will be asked for a computerprocessed text of the final version of the paper, supplemented with illustrations and tables, prepared as a Microsoft Word or LaTeX document. The illustrations are to be presented in TIF, WMF, BMP, PCX or PNG formats (the resolution of point graphics pictures is 300 dots per inch).

Contacts for communication

Valentina Dagienė
Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
Phone: +370 5 2109 732
Fax: +370 52 729 209
E-mail: valentina.dagiene@mif.vu.lt

Internet Address

All the information about the journal can be found at:

<https://ioinformatics.org/page/ioi-journal>

Olympiads in Informatics

Volume 17, 2023

Foreword	1
G. KÉPES, Á. ERDŐSNÉ NÉMETH As the Epitome of Talent: John von Neumann and Hungarian-born Scientists Around Him	3
G. AUDRITO, M. CIOBANU, L. LAURA Giochi di Fibonacci: Competitive Programming for Young Students	19
B. GAÁL The Introduction of Micro:bit in Elementary School, from Unplugged Activity to Programs	33
B. GAÁL Online Robotics Activities During the Pandemic Period – Challenges and Experiences	43
L.G. MENYHÁRT, L. ZSAKÓ Elementary Algorithms – Prefix Sum	53
V. NATALI, NATALIA, C.E. NUGRAHENI Indonesian Bebras Challenge 2021 Exploratory Data Analysis	65
P.S. PANKOV, A.A. BELYAEV Latent and Evident Knowledge to Compose and to Solve Tasks in Informatics	87
T. VERHOEFF Understanding and Designing Recursive Functions via Syntactic Rewriting	99
M. VISNOVITZ, G. HORVÁTH Trends in Teaching Programming in Schools in Hungary	121
REPORTS	
M. DOLINSKY Secondary School Programming Olympiads in Gomel Region	131
F. JINGGA, Y.K. ISAL, A. CENDRANATA, M.I. LIEM, A. MULYANTO Change Management in Preparing Indonesian Team to IOI	143
S.N. KODITUWAKKU, T. GUNAWARDENA National Olympiad in Informatics: Sri Lanka	159
M. OPMANIS, D. SILIŃA, S. SILIŃA, P. PAKALNS Team Competition in Informatics and Mathematics “Cēsis”	173
REVIEWS, COMMENTS	
O. HAZZAN, K. MIKE Guide to Teaching Data Science: An Interdisciplinary Approach	189

Publisher office: Vilnius University

Akademijos str. 4, LT-08663 Vilnius, Lithuania

July, 2023

Olympiads in Informatics

Volume 17, 2023

Foreword	1
G. KÉPES, Á. ERDŐSNÉ NÉMETH As the Epitome of Talent: John von Neumann and Hungarian-born Scientists Around Him	3
G. AUDRITO, M. CIOBANU, L. LAURA Giochi di Fibonacci: Competitive Programming for Young Students	19
B. GAÁL The Introduction of Micro:bit in Elementary School, from Unplugged Activity to Programs	33
B. GAÁL Online Robotics Activities During the Pandemic Period – Challenges and Experiences	43
L.G. MENYHART, L. ZSAKÓ Elementary Algorithms – Prefix Sum	53
V. NATALI, NATALIA, C.E. NUGRAHENI Indonesian Bebras Challenge 2021 Exploratory Data Analysis	65
P.S. PANKOV, A.A. BELYAEV Latent and Evident Knowledge to Compose and to Solve Tasks in Informatics	87
T. VERHOEFF Understanding and Designing Recursive Functions via Syntactic Rewriting	99
M. VISNOVITZ, G. HORVÁTH Trends in Teaching Programming in Schools in Hungary	121
REPORTS	
M. DOLINSKY Secondary School Programming Olympiads in Gomel Region	131
F. JINGGA, Y.K. ISAL, A. CENDRANATA, M.I. LIEM, A. MULYANTO Change Management in Preparing Indonesian Team to IOI	143
S.N. KODITUWAKKU, T. GUNAWARDENA National Olympiad in Informatics: Sri Lanka	159
M. OPMANIS, D. SILIŃA, S. SILIŃA, P. PAKALNS Team Competition in Informatics and Mathematics “Cēsis”	173
REVIEWS, COMMENTS	
O. HAZZAN, K. MIKE Guide to Teaching Data Science: An Interdisciplinary Approach	189