Olympiads in Informatics 16

IOI

International Olympiad in Informatics

ISSN 1822-7732

INTERNATIONAL OLYMPIAD IN INFORMATICS
VILNIUS UNIVERSITY

OLYMPIADS IN INFORMATICS

Volume 16 2022

Selected papers of
the International Conference joint with
the XXXIV International Olympiad in Informatics
Yogyakarta, Indonesia, 7–15 August, 2022

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

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
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, 2022
Vilnius University, 2022
All rights reserved

Olympiads in Informatics, 2022, Vol. 16, 1–2
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.00

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 COVID-19 pandemic is still with us, but – like many international events – we are beginning to explore how we can manage the health risks yet still bring the community back together again. After all, the IOI is a prestigious international competition, but it is also more than that. It is an exchange of cultures and ideas; an opportunity for young students to travel and experience new places and meet new people; an opportunity for team coaches and educators to share the different ways in which they identify and nurture talent in their own countries.

We must acknowledge that the world is not an equitable place. Different countries are moving through the pandemic in different ways, due to a mix of geography, policy, economics, access to vaccinations, public opinion, and of course sheer luck. Not every country will be able to attend IOI in person, and not every speaker will be able to speak live at the IOI conference. For this reason we are enormously grateful to our hosts in Indonesia for committing to a hybrid IOI, where countries who cannot attend on-site are still able to compete online. Unlike some other international events, this ensures that the IOI can start to move out of the pandemic without leaving anybody behind.

The pandemic is of course not the only challenge that we are facing as an international community. The invasion of Ukraine by the Russian Federation has impacted many international events, and the IOI is no exception.

The IOI's response to Russia and Belarus has been to accept online contestants from these countries, but only under a neutral IOI flag, with no national names or symbols. By doing this we aim to take a clear stand against what Russia and Belarus are doing in Ukraine, yet remain engaged with the young teenagers who are the future of these countries, and for whom a genuine engagement with the international community can only help.

We do appreciate that the IOI is a scientific event, not a political one, and that there have been many conflicts in the past where we have remained silent. However, it is also difficult to ignore the scale of the destruction, the widespread attacks upon civilian, medical and educational targets, and the potential for this to spark a greater conflict that is unspeakably worse. While it is impossible to draw a precise line as to when IOI should or should not take a stand on international matters, we have nevertheless decided that this time a line has been crossed.

I am very excited for the Olympiad this year, and for the chance to discuss in person some of the other challenges that our community faces. One of our longest and still most significant challenges is diversity – in particular, gender diversity. To this end it was wonderful to see the first ever European Girls' Olympiad in Informatics hosted by Switzerland last year. I encourage you to keep an eye out for the second edition in Türkiye in late 2022 (`https://ubilo.tubitak.gov.tr/egoi2022/`).

Finally, my thanks to the editors and organisers of this journal, as well as all of the authors who have submitted papers. I look forward to seeing the talks in August in Yogyakarta!

Benjamin BURTON
President of IOI

Olympiads in Informatics, 2022, Vol. 16, 3–11
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.01

Posing Creative Reduction Tasks

David GINAT[1], Shlomit ARIAN[1], Oren BECKER[2]
[1]*Tel-Aviv University, Science Education Department, Ramat Aviv, Tel-Aviv, Israel 69978*
[2]*Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WB, UK*
e-mail: ginat@post.tau.ac.il, shlomit.arian@gmail.com, oren.becker@gmail.com

Abstract. Reduction is a fundamental computer science (CS) notion (Schwill, 1994). In solving reduction tasks one must think at the problem level, in order to recognize suitable correlations between problems. Thinking at the problem level involves recognition of declarative features of problems. This requires a high level of abstraction. Reduction is well apparent in the more advanced stages of CS studies, but it is also relevant at earlier stages. It may serve as an important means for practice and awareness of abstraction. We designed reduction tasks for the earlier tutoring and training of algorithms students. Our designs are illustrated with three creative tasks of different characteristics. Student solutions for each task are presented and discussed. The students demonstrated different levels of abstraction, insight and flexibility in solving the tasks.

Keywords: reduction, abstraction, task design.

1. Introduction

Algorithmic problem solving involves various techniques, among them divide-&-conquer, backtracking, greedy computation, dynamic programming, and means end analysis. One additional approach is that of transformation, sometimes in the form of reduction. Reduction involves mapping from one problem to another based on strong correspondence between the problems. While it is primarily known as a technique for proving intractability of problems (e.g., NP-complete), it is also relevant in algorithmic problem solving. For example, the problem of Maximum bipartite matching is solved by reducing it to the Maximum flow problem, with constructing a network in which flows correspond to bipartite matchings (Cormen *et al.*, 1990).

Reduction is based on concise and efficient mapping from the entities of a source problem S to corresponding entities in a target problem T, so that the solution of Problem-T yields the solution to Problem-S. The example above demonstrates the relevance of reduction with an advanced algorithmic problem, yet it may be as relevant in solving simpler problems. Since the principal theme of reduction is exploitation of problem correspondence, it requires the problem solver to think at the highest, problem level of Perrenet *et al.*'s abstraction levels of algorithm perception (2005). Thinking at the problem

level may deepen one's algorithmic conceptions, which may sometimes tend to focus on the operational "how" of a computation rather than on the declarative "that" of problem features (see (Ryle, 1946) for "how" and "that"). Awareness and practice of the latter are important for competent algorithmic problem solving.

Problem solvers naturally seek problems that are analogous, or similar to their posed problem. Often, they borrow notions, or schemes from other similar problems. When strong analogy in the form of equivalence is recognized, one may borrow another problem's complete solution. The solution may be modified, in adaptation to a given problem, or may be used as a black box, whose inner components are hidden. Reduction encapsulates the latter.

The challenge in reduction involves two phases – Phase-I, of finding a suitable Problem-T to which the posed Problem-S will be reduced; and Phase-II, of providing a concise and efficient transformation of the input of Problem-S to the input of Problem-T. There may also be a need to adjust the output of Problem-T to the output of Problem-S. In Phase-I, the search for a suitable Problem-T requires familiarity with potentially candidate problems, and flexibility upon the examination of correspondence (e.g., noticing that the minimum in a list of negative numbers is the maximum of their absolute-values list). In Phase-II, one should capitalize of Problem-T's features, and develop an elegant and efficient transformation of Problem-S's input to Problem-T's input (and later, possibly adjust Problem-T's output). The processing of the input transformation should not intervene with T's black box computation.

The mapping between problems S and T requires abstraction, in perceiving problems as objects. So is the conception of T's computation as a black box (Perrenet *et al.*, 2005). The focus is on matching structural features of the two problems. Structural matching requires competent pattern recognition (Mayer & Wittrock, 1996; Muller & Haberman, 2008). Armoni *et al.* (2006) illuminated CS student difficulties with these abstraction elements in reductive thinking, among them reduction to the solution, rather than to the problem, and the need to look inside the black box. Ginat and Armoni (2006) showed an example of student difficulties in turning to the notion of complement, when solving the problem of finding a minimum-weight set of edges in a weighted graph, such that each graph cycle has a representative in that set. Students examined graph cycles, rather than turning to a simple reduction.

IOI competitions and training involve problems whose solutions require analogy associations, various transformations, and possibly reductions to other problems. Problem solvers should develop and demonstrate abstraction competencies of thinking at the problem level (in addition to the algorithm level) and matching between structural features. In addition, they should develop creativity in applying mapping between problems, and demonstrate awareness of the importance of sound as well as efficient utilization of black boxes.

We designed throughout the years learning and practice materials for developing and enhancing the above competencies among students, already at early stages of our IOI training. In what follows, we display tasks developed and posed to trainees following our national competition. We also posed some of the tasks to CS students in the second and third years of their undergraduate studies.

The paper illustrates our design and experience with three creative reduction tasks. In the illustrations we describe the design steps and considerations. One design started from a chosen Problem-T, another started from a selected transformation, and a third – from an invented Problem-S. The solution of each task required different flexibility elements. When Problem-S was posed to the students, Problem-T was sometimes provided and sometimes not. We display our experience with students. The student solutions reflect different levels of abstraction and insight into the tasks. The reader may be interested to try solving a task before reading its design description.

2. Task Designs and Solutions

The three tasks presented in this section do not require knowledge beyond searching & sorting and the time complexities of their common algorithms. Each task presentation starts with its design description, continues with the task specification, and ends with our experience with students. In the cases where Problem-T was not provided when Problem-S was posed, students had to demonstrate both phases I and II of the solution process mentioned in the Introduction. When Problem-T was provided, only phase II was relevant. This was still challenging for quite a few.

Arithmetic Shuffle

First, ***Problem-T*** was chosen. The problem input is a list of N integers, and the output is the number of pairs of identical integers; e.g., for the input **5 3 3 2 3 3** the output will be **6**. This problem can be solved in O(NlogN) time by first sorting the list, and then counting the number of identical pairs in the ordered outcome.

Next, a ***transformation*** was chosen. Its output had to be in a format adequate to Problem-T. We chose an N-integer sequence. The transformation was chosen to be: $\langle x_1 \ldots x_N \rangle \rightarrow \langle x_1-1 \ldots x_N-N \rangle$. That is, for each element in the original sequence, the transformation subtracts its location from its value.

Then, ***candidates for Problem-S*** were explored. We sought a natural meaning of identical elements in the transformed sequence. To do so, we wrote the expression for identity of elements explicitly: $x_i - i = x_j - j$. This equation *is equivalent to* $x_i - x_j = i - j$. The new formulation suggested a natural meaning.

> *An identical pair of elements in the transformed sequence (Problem-T's input) corresponds to a pair of elements in the original sequence (Problem-S's input), for which the difference between the values equals the difference between the locations.*

Notice that the above new formulation, of $x_i - x_j = i - j$ may be regarded as an *arithmetic shuffle* of the original relation $x_i - i = x_j - j$.

A ***first attempt of Problem-S*** was formulated.

Given a list of N integers $x_1 \ldots x_N$ *how many pairs are there such that* $x_i - x_j = i - j$*?*

Then, an ***analysis of the first attempt*** was conducted. The naïve, brute-force solution of the formulated problem is to examine each pair of numbers in the sequence. The time complexity of that is $O(N^2)$, which is far worse than the O(NlogN) – the time complexity of applying the transformation and then solving Problem-T efficiently with the transformed list.

Lastly, a ***refined Problem-S*** was designed, to make it more appealing. We replaced the condition $x_i - x_j = i - j$ with the more natural condition $|x_i - x_j| = |i - j|$. The solution of this formulation of Problem-S is slightly more challenging, as one has to properly handle absolute values. Yet, it is based on the same observations, and its time complexity remains O(NlogN). We let the reader complete the analysis of this formulation.

Problem-S. Values and Locations Distances. Given a list of N integers, output the number of pairs of elements in the list, for which the distance between their values equals the distance between their locations.

Example: For the input **6 5 4 1 2** the output will be **7**, – due to the pairs **6** and **5**, **5** and **4**, **6** and **4**, **1** and **2**, **6** and **2**, **5** and **2**, and **4** and **2**.

Problem-T. The number of pairs of identical elements in a list of N integers.

Problem-T was not provided to the students.

A non-negligible amount of students struggled with this task. Quite a few offered the brute-force solution, sometimes with erroneous attempts to avoid some comparisons. Other students simplified the condition of "distance" between the values to "difference", which may be negative. This did help them realize the original arithmetic shuffle specified earlier, where no absolute values are involved. They recognized the relevance of Problem-T and invoked it. Their output was correct in the cases where the results of the subtractions in both sides of the equation $x_i - x_j = i - j$ have the same sign. The better students showed further insight and provided the full answer.

The main challenge here was to represent Problem-S's specification mathematically, and possibly attempt various manipulations in stages – first manipulations when the absolute values requirement is removed, and then when it is returned. One had to demonstrate creative flexibility of the train of thought. Competence in employing the heuristic of simplification, together with flexible manipulations, expressed abstraction in the sense that one did not immediately seek the "how" of the computation, but rather carefully examined the "that" of Problem-S, sought insight into its hidden patterns, and only then looked for a relevant Problem-T and a suitable transformation.

Padding Transformation

First, a ***transformation*** was characterized. In the previous task the sizes of the input and output of the transformation were equal. However, the sizes of the inputs of problems S and T may not necessarily be equal. One should also be acquainted with cases in which

the sizes are different. The idea here was to focus on this notion, without embedding additional challenges.

Next, a ***common transformation feature*** was sought and chosen. Decidability proofs employ the feature of padding when the sizes of the inputs of problems S and T differ. *Padding* is occasionally applied when the input of Problem-S should be augmented. The augmentation may be conducted in different ways. One of them is that of repeatedly adding to the input the same value in a quantity needed, in order to "bring it" to the size of the input of Problem-T.

Then, the ***relation between the inputs*** was defined. The problems S and T may be similar, but differ from one another in a relative value, or position that should be processed. One such case, in which padding may be useful is the following.

> *If Problem-S will compute the i-th largest element in a sequence and Problem-T will compute the j-th largest, and j < i; then Problem-S may be solved by transforming its input to Problem-T, and padding its input in a corresponding augmentation.*

At this stage, ***Problem-S and Problem-T*** were formulated. The described augmentation depends on the values of i and j above. We chose these values to be simple, as the focus is on invoking the notion of padding. The values of i and j were chosen to be N/2 and N/3 respectively.

> *Problem-S will compute the median in an unordered array Arr of N different values, and Problem-T will compute the "thirdian" – the element that is larger than one third of the elements of Arr and smaller than two thirds of the elements.*

Finally, the details of ***Problem-S's input augmentation*** were written and evaluated. An O(N) reduction computation was formulated, as presented below.

> *Find the Max element of Arr, and add to Arr the N/2 values: Max+1, … , Max+N/2; i.e. pad Arr with large values to be 3/2 of its original size.*

The resulting task was the following.

Problem-S. Median. Given an array of N distinct values, output its median.

Problem-T. "Thirdian". Given an array of N distinct values, output its "thirdian", which is the element that is larger than one third of the elements and smaller than two thirds of the elements.

Problem-T was provided to the students.

We posed the task to a limited group of students. Unfortunately, padding solutions were not offered. The students turned to reduce the size of Arr. The main theme that was demonstrated was the removal of elements smaller than the median of Arr, one by one. One removal version involved a utilization of Problem-T as an operator, with repeated calls for the removal of single "thirdians", one at a time. This reflects a degenerated transformation and exploitation of Problem-T.

Another version involved sorting of Arr, and the removal of an amount of the smallest elements of Arr that will "shift" the median to the "thirdian" position. Students erred with the correct number of removed elements. And, obviously, the computation complexity exceeded O(N).

It seems that students demonstrated different kinds of impasse. Perhaps their lack of experience with the heuristic of auxiliary construction hindered them from choosing the direction of padding the original input. They followed a direction of "in place" computation, with reduction of Arr's size. In addition, Problem-S involved the notion of median, and this may have led some in the direction of ordering calculations, even at the cost of a very inefficient solution. Thinking at the problem level of Perrenet *et al.* (2005) was very limited, and there was no capitalization on the similarity between the two problems for providing a transformation that yields a single, elegant reduction to Problem-T.

Location-Value Relation

First, ***Problem-S*** was designed. A special case of a previously invented problem – the Widest inversion (Ginat, 2008) – may be solved in a simpler way than the original, general problem. The input of the Widest inversion problem is a list of N positive integers, and the output is the largest distance between two unordered integers in the list; e.g., for the input **2 5 4 6 3** the output will be **3**. The solution is not that simple.

In the special case where the list is a permutation of the integers 1 to N, an elegant solution may capitalize on the particular property of a permutation, which is:

> *When the input is a permutation of 1..N, the range of values is exactly the range of locations.*

Next, we sought a ***transformation***. We examined a simple example, and looked at the possibility of transforming a given permutation, such as **2 5 4 1 3** to a list of the locations of the permutation values: **4 1 5 3 2** (e.g., the 1-st value in the new list is **4** since **1** appears in the 4-th place in the original permutation). Upon looking at these two lists one may notice the following:

> *The "Largest drop" – the largest difference between two unordered integers – in the new list is the widest inversion in the original list.*

Thus, if the Largest drop is a simple problem it may become ***Problem-T***. In an **analysis** of this problem, one may notice that the computation is simple – an O(N) time, of one "pass" over the input, where the difference between every newly read value and the current Max is examined. We obtained the following task.

Problem-S. Permutation Inversion. Given a permutation of the integers 1..N, in an arbitrary order, output the largest distance between two unordered integers.

Example: For the input **1 6 2 4 7 5 3** the output will be **5**, which is the distance between **6** and **3**.

Problem-T. Largest Drop. Given a permutation of the integers 1..N, in an arbitrary order, output the largest difference between two unordered integers. The output in the above example will be **4**. This difference occurs twice – between **6** and **2**, and between **7** and **3**.

Problem-T was sometimes provided to the students.

When Problem-T was not provided to the students, they offered two kinds of solutions to Problem-S – a brute-force $O(N^2)$-time solution, in which the distance between every pair of integers is checked; and an insightful O(N) solution which capitalizes on the observations italicized in the above design.

When we provided Problem-T with Problem-S, and requested a reductive solution, some students indeed offered the above elegant reduction. However, others still did not see the correspondence between the problems, and turned to a brute-force solution. Since they were obliged to solve by reduction, some of them used Problem-T as an operator which receives only a pair of values. Their solution called Problem-T's (black box) algorithm $O(N^2)$ times, a separate call for each pair examined by their brute-force solution.

Some created a list of pairs <i,j>, such that i is greater than j, and is the furthest location of an integer smaller than the integer whose location is j, in the original input. For example, for the input in Problem-S's specification, the list of pairs will be the following: <7,2>, <7,4>, <7,5>, <7,6>. (The 7 in all the pairs is due to the location of 3; the 2 in the first pair is due to the location of 6; the 4 in the second pair is due to the location of 4; the 5 in the third pair is due to the location of 7; etc.) Each pair was computed separately, thus the time complexity is $O(N^2)$.

Both of these inefficient solutions express a "reduction by obligation". The first, "operator based" solution utilizes a degenerated variant of Problem-T's solution, and demonstrates limited abstraction at the problem level. The second solution expresses a slightly higher abstraction by invoking Problem-T's solution only once, but lacks sufficient insight of the correlation between the problems.

3. Discussion

Reduction is a fundamental CS notion. Although it is mostly apparent in advanced courses, it may be a relevant tool also in the Introduction to Algorithms level. It requires recognition of patterns, creativity, and thinking in the problem level. As such, it may be introduced and practiced by IOI students rather early in their training.

The practice of seeking problem correspondence in reduction, as well as applying it properly and efficiently, develops one's analogical thinking and enhances awareness of essential algorithmic problem solving elements, including moving between different levels of abstraction (the problem level and the algorithm level), revealing underlying patterns, and employing flexibility is developing suitable algorithmic schemes.

The first task required the recognition of a mathematical underlying pattern. Although the pattern was simple, one needed flexible manipulations to reveal it. This was

also relevant in the third task, where the underlying pattern was a simple location-value pattern. In the second task one had to demonstrate flexibility in turning to a concise, elegant construction. All the tasks involved the application of problem solving heuristics – problem simplification in the first task, auxiliary construction in the second, and a change of representation in the third task. In addition, all the tasks required thinking at the more abstract problem level, both upon seeking problem characteristics and upon mapping from Problem-S to Problem-T.

Students demonstrated various levels of the above. Many did not recognize underlying patterns, expressed limited flexibility, and did not properly relate to efficiency considerations. In addition, some students did not fully capitalize on Problem-T's characteristics. Some invoked its solution repeatedly as an operator, thus demonstrating a degenerated transformation that "misses the point" of reduction. We believe that practice and awareness play a key role in developing suitable, desired competencies. Such a development will help assimilating abstraction, which is one of the most essential elements in computer science and computational thinking.

References

Armoni, M., Gal-Ezer, J., Hazzan, O. (2006). Reductive thinking in computer science. *Computer Science Education*, 16(4), 281–301.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. (1990). *Introduction to Algorithms*. MIT Press.

Ginat, D., Armoni, M. (2006). Reversing: an essential heuristic in program and proof design. In: *Proc of the 38th ACM Computer Science Education Symposium - SIGCSE*. ACM Press, 469–473.

Ginat, D. (2008). Learning from wrong and creative algorithm design. In: *Proc of the 40th ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 26–30.

Mayer, R.E., Wittrock, M.C. (1990). Problem-solving transfer. *Handbook of Educational Psychology*, 47–62.

Muller, O., Haberman, B. (2008). Supporting abstraction processes in problem solving through pattern-oriented instruction. *Computer Science Education*, 18(3), 187–212.

Perrenet, J., Groot, J.F., Kaasebrood, E. (2005). Exploring students' understanding of the concept of algorithm: levels of abstraction. *ACM SIGCSE Bulletin*, 37(3), 64–68.

Ryle, G. (1946). Knowing how and knowing that. In: *Proc of the Aristotelian Society*, 46, 1–16.

Schwill, A. (1994). Fundamental ideas of computer science. *Bulletin of European Association for Theoretical Computer Science*, 53, 274–295.

D. Ginat – served as the head coach of Israel's IOI project in the years 1997–2019 (team leader in 1997–2007). He is the head of the Computer Science Group in the Science Education Department at Tel-Aviv University. His PhD is in the Computer Science domains of distributed algorithms and amortized analysis. His current research is in Computer Science and Mathematics Education, with particular focus on various aspects of problem solving and learning from mistakes.

S. Arian – received her M.Sc. in Computer Science from The Academic College of Tel Aviv-Yaffo. For the last 15 years she is teaching various computer science courses, including Algorithms, Data Structures and Computability Theory. Her current research is in computer science education, particularly about abstraction facets.

O. Becker – served as Israel's Team Leader for the IOI in the years 2009-2014. He is a postdoctoral researcher at the Department of Pure Mathematics and Mathematical Statistics at the University of Cambridge. His PhD connected geometric and measurable group theory to the computer science domain of property testing. His current research is, in addition, on expander graphs, word maps and random groups.

Olympiads in Informatics, 2022, Vol. 16, 13–22
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.02

How Competitions Can Motivate Children to Learn Programming

Aliya KATYETOVA
Doctoral School, Faculty of Informatics, Eötvös Loránd University
Budapest, Hungary
e-mail: akatyetova@inf.elte.hu, katyetova@mail.ru

Abstract. In the present time, more and more children easily play with LEGOs, labyrinths, and puzzles, which help to master basic logical skills and algorithm creation. Playing the games they compete with peers at home and in a primary school to be faster and assemble a construction correctly, to set up and be able to manage a drone or a toy robot. Children are enthusiastic about competing and getting ahead of each other and with pleasure want to know how a drone works, how a robot is programmed, how a joystick is configured for a game. This article aims to provide teachers and parents with the evidence how competitions can motivate children to learn programming.

Keywords: competitions, motivation, children, programming.

1. Introduction

Motivation of students to learn is important for both teachers and parents. Competitions offer a convenient way to bring informatics concepts to students in a more different fashion than traditional teaching in schools. It could be said that the tasks are the heart of the competition. Therefore, designing tasks that support the goals of the competition is an important and demanding undertaking.

Nowadays, there are many different informatics competitions from small to worldwide events (Scratch Olympiad, IOI, Bebras, etc.). Moreover, the types of tasks vary from easy ones solved with a pen and some paper to complex problems dealing with large datasets and sophisticated algorithms. Many different types of events offer a wide range of possibilities for pupils to get involved in informatics.

Competitions can encourage students who want to learn informatics in depth. They can also be a source of motivation for students interested in programming (Dagienė, 2010).

2. The Importance of the Competitions

According to Manev *et al.* (2009), it is necessary to teach students to start to compete as early as possible. In many countries, there is a large gap between the knowledge and skills acquired through the regular curriculum and those required for informatics and algorithmic contests and competitions. Thus, teaching programming as well as preparing for informatics contests is basically an activity outside a school. There is a need for more independent work on the part of students and a more specific attitude from teachers and parents (Nemeth and Zsako, 2018).

Children at school and home became more interested in learning how to create new information communication technologies. They are easy to play LEGO, independently assemble mazes and puzzles that help them master the basic logical skills of creating algorithms. Competing with each other, children curiously examine robots, drones, joysticks, and other automated toys. They with burning eyes and interest ask their peers, parents, and teachers how they are arranged and how they work. Therefore, when schools announce the call of competitions and Olympiads in informatics for children, they willingly participate in them.

In the article "Kids Programming Marathon: A Step toward Better Engagement with Computer Science Education" (Taki and Alnahhas, 2019) authors state that computer science education is an extensive subject. It combines all the "STEM" subjects: science, technology, engineering, and math, and also includes design. It is important for students to master these skills because computer science is everywhere in our life. By expanding access to computer science for all young people as early as possible, it is necessary to help them prepare for current and future work. This education provides them the opportunity to become the next world innovators and programmers (Taki and Alnahhas, 2019).

Competitions are one of the essential ways to encourage pupils to discover new fields. They introduce children to various fields of science, including informatics, where the emphasis is on testing problem-solving and logical analysis skills through an exciting experience in which the participant learns programming, thereby satisfying their interests. Therefore, many countries are working on organizing informatics and also programming competitions for children, in order to develop not only the skills of creative thinking for pupils but algorithmic thinking as well. Furthermore, it is important for improving motivation to learn computer programming.

In addition, while writing this article the author wondered why many countries have informatics competitions for children, while some do not have such contests, particularly in programming for primary schoolchildren. At the same time some nations (Syria, etc.) are working on organizing a programming marathon for children and adolescents to increase the general scientific level and enhance the skills of analysis and creative thinking among pupils (Taki and Alnahhas, 2019). In Kazakhstan, competitions for children are held only from grades 7–8. It is a challenge to think and start organizing informatics contests by engaging primary schoolchildren. Such contests will make it possible to interest them, even more, to study computer science outside school, to learn programming with the help of educational games.

3. The Power of Motivation

The primary school years are important in children's lives. In this formative period, boys and girls profoundly affect their mental and emotional growth. Today's schools are challenged to provide meaningful experiences that will help these children realize their full potential. And in this case, motivation plays a huge role.

What does motivation give to children? First of all, it is a good tool for education and development. Then there is the importance of computer science education. These are also new academic achievements and increased interest in the subject.

When children have a motivation they can come up to a number of things, including:

- Self-confidence.
- Lack of fear.
- Focusing.
- Comprehension.
- Good organization skills.

While some students are natural self-motivators, many children struggle to find the motivation needed to do their best. Therefore, competitions can help in this situation. At the same time teachers also help them in informatics classes.

3.1. *How can Competitions Motivate Children to Learn Programming?*

The goal of all teachers is to help pupils become self-motivated students. And competitions provided in the computer classes and/or online via Zoom, Microsoft Teams or Skype help them it this deal. To do this, it is important to involve parents in playing the games together with their sons and daughters which builds motivation. It can be done at a time convenient for parents, in the evenings or on weekends. All of these things help children to improve confidence and motivation to learn.

For example, competitions in Informatics organized by schools include different Informatics and ICT themes starting from easy tasks to difficult ones. It depends on whether the junior or senior pupil is participating and what the purpose of this event will be.

The following tasks features should be taken into account (Hakulinen, 2011):

- The problem should be clearly formulated.
- The tasks should be easy to understand.
- The algorithms solving the problem may be a modified version of the classical algorithm.
- There should be several different acceptable solutions of varying complexity and efficiency.
- The result should be clear and concise (depending on the complexity of the task).
- Tasks might be interactive and using questions.
- Short non-programming tasks can be used to attract new students.
- Other criteria.

These tasks provide pupils with logical and algorithmic thinking development. Algorithmic thinking is influenced by many human cognitive factors. This means not only abstract and logical thinking but also creative abilities and problem-solving competences, as well as the ability to think in structures. This complexity creates difficulties in learning and developing the algorithmic thinking of pupils. We need to reduce the complexity to the level, where the concepts of algorithmic thinking can be learned in a natural and playful way. The following is recommended: use of basic actions, natural description language for writing the algorithms, and an interactive environment with possibilities for experimentations also flexible for a run variety of the algorithms. The problems to be solved must be adequate to the pre-knowledge of the children-beginners.

And competition here is a good motivation tool for learning, particularly in programming.

In this part of the article, we can cite the following example from an article by Kubica and Radoszewski (2010), who proposed using tasks that require algorithmic thinking, but not programming, in order to attract novice students who know nothing about programming or algorithms. They claim that offering tasks or puzzles that require different levels of algorithmic thinking is a good way to popularize programming learning among younger schoolchildren. They also proposed a couple of tasks that require algorithmic thinking but are formulated in a purely mathematical way. The problems are designed in such a way that the desired solution minimizes the total time for its manual execution.

4. Learning Computer Programming

Being primary school students, children develop an interest in academic subjects, identify inclinations to various fields of knowledge, types of work, develop moral and cognitive aspirations. However, this process does not occur automatically, it is associated with the activation of cognitive activity of students in the learning process. One of the effective means of developing cognitive interest in computer science lessons in primary school is a game.

For a child of primary school age, playing is of the utmost importance: for them it is study, work, a serious form of education.

The inclusion of games and game moments in the lesson makes the learning process interesting and entertaining, creates a cheerful working mood for children, facilitates overcoming difficulties in mastering the educational material. Another positive side of a game is that it promotes the use of knowledge in a new situation, thus, the material assimilated by younger schoolchildren goes through a kind of practice, brings variety and interest to the educational process.

Playing is a natural and humane form of learning for a child. By teaching through games, we teach children not how it is convenient for us, adults, to give educational material, but how it is convenient and natural for children to receive it.

There are various types of games that contribute to the development of cognitive interest of younger schoolchildren. These are exercise games, competition games, story-role-playing games, educational travel games, etc.

The majority of students show an interest only when the lesson's topic is interesting to them or the teacher uses unusual teaching techniques, particularly, a game. A teacher should stimulate and develop the cognitive interest of younger students in each computer science lesson. It is necessary to strive to ensure that most of the class has a high level of cognitive interest, that is, that primary school students are active in every lesson. To do this, a teacher needs to include game elements or game situations in the structure of each computer science lesson and in informatics competitions.

4.1. *Teacher Engagement*

One-to-one programming is probably the best form of teaching programming because the teacher can focus on one student and promptly give feedback and correct the student's misunderstandings. However, in real conditions, one teacher often teaches ten, twenty, or even more students at the same time. In this case, quizzes and tests with appropriate feedback can help to identify misunderstandings of the students about programming concepts and to correct them (Brown and Wilson, 2018).

Using live coding can also help students. If the teacher creates a program in front of their students instead of using and showing presentation slides, the teacher can react more sensitively to students' "what if?" questions. Also, learners can see that making mistakes during programming is normal and they can learn to find and correct these errors. In addition, a teacher may ask their students several times during a live coding lesson to predict the results before the application is executed (Brown and Wilson, 2018). Despite the fact that live coding might be slower than using prepared slides, Stoffova, V., and Vegh, L. believe that it is worth taking the time to try at least a few times during a programming lesson (Stoffova and Vegh, 2019).

Pair programming, when two students share one computer, is also a good practice in teaching computer programming. One of the students types the code, while the other comments, prompts a classmate and makes suggestions. It is important to change the role several times per lesson. During pair programming, students can help each other, they can explain each other's misconceptions (Brown and Wilson, 2018).

It is also important that teachers give students motivational tasks that interest them. Since almost all students like computer games, it can be a good choice to assign learners to create some simple games. A computer game as a project adds an element of fun to programming, captivates children, and they stay interested from the beginning to the end (Doherty and Kumar, 2009). However, because children in primary school are not proficient in programming, it would be difficult for them to write a complex computer game from Scratch. Framework-assisted computer programming, where children use a prepared application framework for creating games, might be an effective solution to this problem. Using interactive online e-learning platforms that merge the possibility of learning programming skills while building digital games (Ivanova, 2016) can be a good solution, as well.

In this case, the Scratch programming environment can help.

4.2. *Scratch as a Motivation Tool*

Scratch as a visual environment for programming and creating games is being used successfully to teach programming skills to novices. It is used in primary and secondary schools in Kazakhstan and other countries as well. Scratch is freely available and easy to install and use for teachers and students. Scratch teaches computational concepts to students in a fun and engaging way. The student engagement with Scratch is superior to the level of engagement while studying ICT literacy skills and motivation levels are also high.

For instance, in Scratch, you program largely visually: Programs are put together from colored building blocks, which allow children at any performance level a very easy entry into programming. At the same time, Scratch can also introduce advanced programming concepts such as object orientation, concurrent processes, or event handling in a very natural way. Scratch is, therefore above all, a tool for conveying important and cross-programming concepts. Verifiable learning objectives are important to us. The acquired knowledge can be applied or expanded in (partly further) tasks directly following the material. The successful completion of the assessment tasks (in the form of a game) at the end of each learning unit guarantees that the essential concepts have been understood. In this case, the children are happy, perform tasks with interest, compete with each other to see who is more beautiful and performed the algorithm better or made an easy game according to the template by themselves.

Teachers can include tasks that have some initial code, which have to be improved. Here Scratch can be used to motivate younger students to participate in a competition and include tasks with some initial game elements that needed to be improved.

Often the games chosen for the competition may not have a known ideal solution. This is how they encourage competitors to think and develop their own original ideas, rather than implement well-known algorithms.

4.3. *Working in Tynker Platform*

Tynker is a creative computing platform that helps kids develop computational thinking and programming skills in a fun, intuitive, and imaginative way. As they are guided through interactive game-based courses, kids quickly learn fundamental programming concepts. With Tynker every child can apply their coding skills as they build games, tell stories, create apps, control drones and robots, and more. The platform even offers a parent dashboard where mothers and fathers can follow their child's success and share their creations.

This opportunity is especially interesting for those children who already like to play with LEGO. They will be more genuinely excited about the opportunity to integrate Tynker with these interests, expanding their potential for games as they learn.

The significant advantage of this platform is the possibility of working in it for primary school students aged 5–10 (Fig.1).

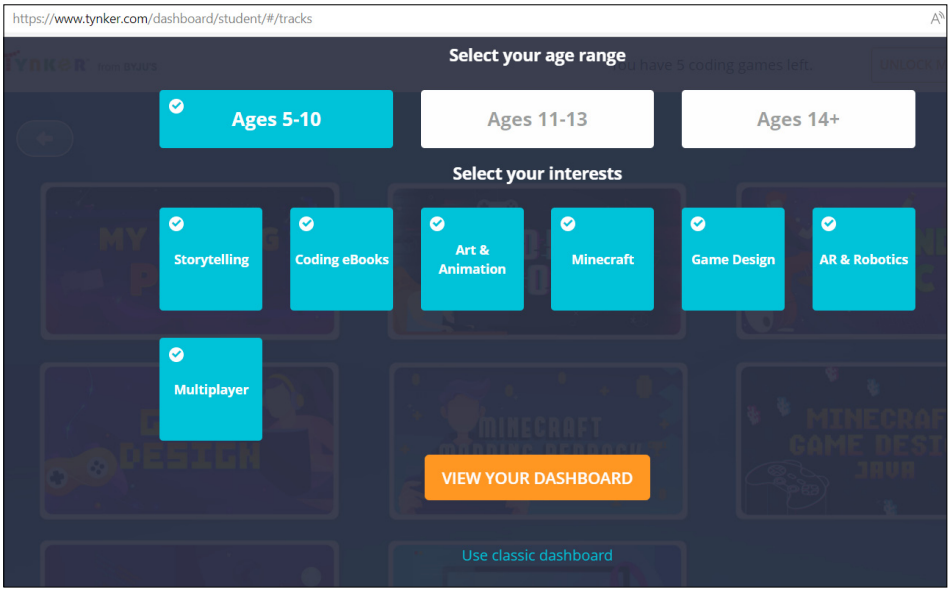


Fig.1. Age range of participants.

Children begin programming using Tynker's block-based visual language, which helps them recognize patterns and master programming concepts such as sequence, loops, conditional logic, and algorithmic thinking. They can show their creativity by animating their games and telling stories using code.

4.4. *Involving into Playing Algorithms*

There are several ways to engage children to playing algorithms not only using Scratch and Tynker platform but also other computer applications.

Understanding algorithms is one of the difficulties that students met when starting to learn programming. For beginners in programming, it is important to understand the principles of algorithms along with the ability to find or create their own algorithms for new problems and tasks. An algorithm prescribes exactly what to do in the given possible situations. This is one of the main educational goals that beginners should know. So, the algorithm manages all possible situations, it is a sequence of steps that leads to the result. It is a possibility to experience in a game that exactly follows such steps of a self-developed algorithm (Czakoova, 2020).

Children understand the principles of algorithms by solving different tasks in competitions organized by schools. Animations of algorithms are used. According to Vegh and Stoffova (2016) in these approaches, the students play algorithms. In this way they get a better understanding of given algorithms.

In the article "Developing algorithmic thinking by educational computer games" Czakoova (2020) suggests an idea of involving students into playing algorithms. The task of the teachers is to motivate the students to improve their algorithms whilst finding more efficient solutions. An example of a computer game in which there are three levels of difficulty to find an effective solution is given. The students take on the roles of a figure and a navigator. They can determine the progress of the game using algorithms. In this way, students will learn more about basic algorithmic thinking. While the students find good solutions, they can learn a lot about sequential algorithms. It is a form of explorative learning, where the students can test algorithms by playing. The goal is to bring fun, pleasure, and motivation to the programming learning process by using games. All this can be used at computer science Olympiads as assignments.

A much greater motivation arises when students get the opportunity to invent their own algorithms to solve a specific problem. A necessary condition is the correct choice of tasks that need to be solved. The best option is to gamify the problem. Gamification opens the way to the introduction of game elements in a non-game situation. Such an educational approach at competitions motivates students to learn using game design and game elements in learning environments. The competitive spirit during the competition and games should inspire children to continue learning new and interesting things. Games in any form increase motivation through engagement. All this is observed as more and more important in education. The goal is to maximize fun as well as engagement by attracting student interest.

Problem methodology is effectively used in a playful way in competitions for the development of algorithmic thinking of primary school students. Students should be directed to become creative problem solvers, experimenters, and creators of alternative solutions. By playing games, the student receives immediate feedback so that he/she can correct his/her actions to get the right solution (Czakoova, 2020).

For novices in programming, educational computer games with the help of programmable toys are suitable tools for teaching the basics of programming. They will be highly motivated by the educational computer game to learn programming (Stoffova and Czakoova, 2019).

The situation is quite different when students have to program a game. This motivation towards games could be used to promote informatics competitions for children and also to get students interested in algorithms. As they learn, children create mini-games, solve puzzles, create programming projects, earn exciting badges (prizes) and discover new characters. That's why kids love to learn by playing – even though they are learning important programming concepts, they feel like they are just playing a game.

5. Conclusion

In this paper from all the mentioned above, it is obvious how important the role of the competitions is for the student's motivation to learn computer programming. They do not only involve children in an interesting and exciting environment but also develop a competitive and team spirit in children to complete tasks. It also helps to instil interest

in studying further topics in computer science and programming, because children are already familiar and know that it will not be boring, but fun and useful.

In competitions, the use of games and game situations using a computer and in the process of teaching computer science is also appropriate and relevant at the stages of programming training.

For many countries, it will be meaningful and useful to think about organizing competitions for younger schoolchildren, especially to engage kids from grades 2–4. Children's computer science competitions will be promising competitions that should be organized where they are not held and supported in order to promote computer science education, as well as for the future development of children's digital literacy.

References

Brown, N.C., Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS Computational Biology*, 14(4), e1006023.

Czakoova, K. (2020). Developing algorithmic thinking by educational computer games. *eLearning & Software for Education*, 1.

Dagienė, V. (2010). Sustaining informatics education by contests. In: *International Conference on Informatics in Secondary Schools-Evolution and Perspectives*. Springer, Berlin, Heidelberg, pp. 1–12.

Doherty, L., Kumar, V. (2009). Teaching programming through games. In: *2009 International Workshop on Technology for Education*. IEEE, pp. 111–113.

Hakulinen, L. (2011). Survey on informatics competitions: Developing tasks. *Olympiads in Informatics*, 5, 12–25.

Ivanova, S. (2016). Learning computer programming through games development. In: *The 12th International Scientific Conference "eLearning and Software for Education"*. "Carol I" National Defence University Publishing House, Bucharest, Romania, pp. 492–497.

Kubica, M., Radoszewski, J. (2010). Algorithms without programming. *Olympiads in Informatics*, 4, 52–66.

Manev, K., Sredkov, M., Bogdanov, T. (2009). Grading systems for competitions in programming. In: *Proceedings of the XXXVIII. Spring Conference of the Union of Bulgarian Mathematicians*.

Németh, Á.E., Zsako, L. (2018). Grading systems for algorithmic contests. *Olympiads in Informatics*, 12, 159–166.

Stoffová, V., Végh, L. (2019). Learning object-oriented programming by creating games. In: *The 15th International Scientific Conference eLearning and Software for Education* (Vol. 1, pp. 20–29). "Carol I" National Defence University.

Stoffová, V., Czakóová, K. (2019). A Playful form of Teaching and Learning using Micro-World Applications. In: *The International Scientific Conference eLearning and Software for Education* (Vol. 1, pp. 110–115). "Carol I" National Defence University.

Taki, M., Alnahhas, A. (2019). Kids programming marathon: A step toward better engagement with computer science education. *Olympiads in Informatics*, 13, 225–235).

Végh, L., Stoffová, V. (2016). An interactive animation for learning sorting algorithms: How students reduced the number of comparisons in a sorting algorithm by playing a didactic game. *Teaching Mathematics and Computer Science*, 14(1), 45–62.

Websites

Daryn.kz – РҒПО «ДАРЫН»
https://www.tynker.com
iBobor

A.D. Katyetova, doctoral student of the Faculty of Informatics of Eötvös Loránd University, Budapest, Hungary, Master of Informatics. She is the author of scientific and methodological papers and articles in Kazakhstan on the informatization of education, distance learning, and children's development. She is the author of the educational and methodical manual for the course "Fundamentals of algorithmization and programming".

She was doing a research internship on the program "Improving University Leadership, Management and Teaching and Learning" in the Reading University, UK, under the "Bolashak" International Program of Kazakhstan (2013). She has been awarded a Stipendium Hungaricum Scholarship (2021).

She worked in the Strategic department in a Kazakhstan educational company, which is a National operator of WorldSkills Kazakhstan competition.

Her research interest includes didactics and methodology of teaching informatics (digital literacy) in primary school, teaching programming languages and databases.

Her current research interest is the digital literacy development of primary schoolchildren.

Olympiads in Informatics, 2022, Vol. 16, 23–34
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.03

Common Approaches to Developing Extensible E-learning Systems

Bojan KOSTADINOV[1], Irena STOJMENOVSKA[2]
[1]*Cloud Solutions LLC*
st. Jurij Gagarin 33/27 Skopje, Macedonia
[2]*University American College Skopje*
Treta Makedonska Brigada 60, Skopje, Macedonia
e-mail: bojankostadinov@gmail.com, irena.stojmenovska@uacs.edu.mk

Abstract. Education plays an important role in all of our lives, as it allows people to reach their potential, develop problem-solving skills, and create more opportunities for employment and self-dependency. Many countries around the world are investing heavily in schools and universities, as an educated population is linked to stronger economic growth, happiness, stability, and a reduction in poverty and crime. One of the best ways to provide equal opportunities for good education is through the use of technology and multimedia. In this paper, we will provide a broad overview of e-learning systems and their most common features, and discuss common patterns and approaches for maintaining and extending them, as well as how to improve their performance and stability using the latest available strategies and technologies. Additionally, this paper features several use cases of such systems, and examples of how they were maintained, extended and improved over time, while making sure privacy concerns and usability are not negatively affected.

Keywords: e-learning, extensibility, web-based systems, STEM education.

1. Introduction

The knowledge, skills and motivation of workers is one of the key factors for both business, regional and country growth. Jobs that have a higher barrier to entry for new employees tend to pay higher salaries, offer more worker benefits and other incentives – while, on the other hand, jobs with a lower barrier to entry (such as those that don't require education or training degrees) tend to offer much less benefits and lower salaries, as companies are able to fill those positions much more easily (Janjua *et al.*, 2011). Offering better primary and secondary education tends to allow poor and underdeveloped countries to bring people out of poverty, and leads to stronger and more sustainable economic growth (Appiah, 2017). Similarly, companies are investing heavily

in specialized education for workers, in order to stay in front of the competition. With automation and AI, it is expected that education will play an even bigger role in the future, as the skills demanded by the labor market will undoubtably change. To summarize, education is one of the most powerful and proven instruments for increasing economic growth, reducing poverty, and lowering inequality.

E-learning systems allow governments, companies and educational institutions to share knowledge with the help of modern devices and technology – which might include sharing textual learning materials, animations, multimedia, and other content forms. Because e-learning systems might be accessed remotely, and are usually web-based, they are one of the main methods for allowing students of all backgrounds to have similar opportunities to access quality education. Most of the time, e-learning systems are used in addition to traditional on-site learning, but some organizations offer students the ability to complete an entire curriculum online, which might include earning a degree or getting a certificate for completing a course or specialization. Some additional advantages of e-learning are consistency, up-to-date content, quick delivery, personalization, analytics, cost effectiveness and better time management.

One example of a popular e-learning system is Moodle, which is an open-source learning platform that is available in several different languages, and which can be used in both primary schools, secondary schools, universities and other companies and establishments. It can be easily installed and hosted on a web server, and enables different ways of sharing content, as well as offering students various features such as a personalized dashboard, calendars, forums, wikis, quizzes, notifications, file sharing, and much more. One of the main benefits of using Moodle is the rich plugin ecosystem (i.e., its extensibility), which allows anyone to add additional features to Moodle, such as new integrations with other systems, gamification, activities, new types of quiz questions or certificates. Other popular and accessible software, such as the content management system WordPress, which can be used for educational purposes as well, also offers administrators the ability to extend the system with plugins and themes.

In this paper, starting with section 2, we will outline several common approaches to developing extensible e-learning systems such as Moodle, how to maintain and improve them over time, and how to keep libraries and tools up to date in order to make sure the systems are performant and secure. Extensibility is a key feature of any software system, and there are several common theoretical strategies and approaches which apply to all systems (including e-learning systems), as well as several differences which must be taken into account.

Next, in section 3, we will provide three use-cases and examples of existing e-learning systems, and how they have been maintained and extended over a period of several years – while making sure they remain performant, stable and usable in different usage scenarios (for example, students using modern web browsers from home, and connecting from outdated browsers in schools). All of the examples are e-learning systems which are used in Macedonia, and all of them have a large userbase of students and teachers. At the end of the paper, we will summarize our findings.

2. Developing Extensible E-learning Systems

Extensibility is one of the key software engineering principles, which corresponds to enabling and planning for future growth and updates. It can be thought of as a metric of the difficulty involved with changing existing features or implementing new ones. Extensibility is also an important consideration when discussing system security and reusability – as we would like to be able to easily implement fixes for newly discovered vulnerabilities and requirements.

A software system will go through different phases in its lifetime, including the initial development, evolution, maintenance, phasing out and termination. Different strategies might be employed to extend the life of a software system (Lavelle, 2005), including waiting (requires no changes), wrapping with middleware, renovating into a more modern form (for example, updating the UI), replacing with a new system which is more up-to-date or offers more features, and finally outsourcing the maintenance or development of such a system outside the organization (so resources can be focused elsewhere).

Such approaches are applicable to e-learning systems as well. Because of privacy concerns (which are stricter when it applies to pupils and students), rules around tracking and data collection, or security and network limitations, specific care must be taken when integrating an e-learning system with new software or outsourcing features elsewhere. There are various rules with regards to storing student data, and some organizations or schools might have Internet limitations that only allow certain domains or IP addresses to be available in classrooms. For example, in e-learning systems we often need to store content such as pdf or video files, but deciding to move the storage of such files to a cloud offering such as S3 in Amazon Web Services might lead to a different set of problems. Similarly, some schools might have specific licensing agreements for operating systems or use outdated browsers, so proper planning and testing is required when deciding to switch to a completely different software system. What might work for one organization or school, might be prohibited at another, so it's best to make sure that the system can be used independently (or, alternatively, that it depends on as little outside services as possible), or that all restrictions are known beforehand.

The first major consideration when discussing extensibility is understanding the different forms that exist, and naturally these will also apply to e-learning systems: black-box extensibility, white-box extensibility, and gray-box extensibility. Black-box extensibility refers to systems where knowledge about the internal system implementation is not important when implementing new features or extensions (only a specification regarding interfaces is used). In contrast, white-box extensibility refers to extensibility where we are modifying the source code of the system. Gray-box extensibility is a compromise between the two.

The second major consideration is related to the source code, where we need to make sure that we follow appropriate principles and patterns to make the code understandable, testable, and easily editable. **SOLID** is a set of five design principles associated with

the quality of object-oriented programming, which is not only related to writing understandable software, but also one which is flexible, extensible, and maintainable. These principles include:

- The single-responsibility principle.
- The open–closed principle.
- The Liskov substitution principle.
- The interface segregation principle.
- The dependency inversion principle.

In short, the principles indicate that there shouldn't be multiple reasons for a single class to change – i.e. a class should have only one responsibility; that software entities should be open for extension but not modification; that derived classes should be usable in place of base classes; that multiple client-specific interfaces are a better alternative than one general-purpose interface (to decrease coupling between systems); and finally, that we should depend on abstractions vs concretions – i.e. objects should not create their dependencies (i.e. we should pass them).

One common approach to extensibility is using plugins. For example, in WordPress, this is accomplished through hooks (actions or filters), which are a way for a function in one piece of code to interact or trigger actions elsewhere – at pre-defined spots. Actions execute code and return without passing anything back, while filters modify and pass values back to be used later in the code. A good extensible system is one where the internal structure or architecture doesn't change (or is minimally affected) by the addition of new functionality.

The third major consideration is related to following a proper project management structure (Yagel, 2017), which includes using version control, documentation, issue databases and API guidelines.

Finally, the fourth major consideration is connected to testing. **Automation testing** is a process that involves an execution of test suites that ensure the quality, security, and performance of a software system. This can be accomplished through different forms of tests, including graphical user interface tests and API tests. Correct use of tests can lead to the development of software which contains less bugs, is more performant, and (of interest to us in this paper) software which is more extensible. Tests enable us to detect issues early in the development cycle (of the software itself, or the development of a new feature), supports easier debugging and enables catching regressions early on – allowing for quicker development and faster releases of new versions. There are different types of tests, including unit tests, integration tests, smoke tests, regression tests, performance tests, etc. End-to-end testing solutions, such as (for example) Cypress – a powerful automation tool for testing any application that runs in a browser, guarantee that changes don't break existing features for users by testing the system in a similar way that a user would. Popular software solutions used in automation testing allow us to execute parallel tests using different platforms, operating systems, and browsers – thus simulating various environments and restrictions that might exist at different organizations (such as, for example, in schools).

3. Examples

In the previous section, we have discussed some important topics that relate to the development of extensible systems – which include proper planning, testing, modularization, architecture, and code structure – all of which are crucial when we are planning to make modifications to any software system, as it might affect its usage, quality, or performance. In this section, we will describe several examples of existing e-learning systems. For all of the outlined systems, the authors of the research paper have been involved either with building or maintaining them. For each system, we will provide a brief overview, describe where it is being used, and discuss the challenges related to maintaining and extending it.

3.1. *MENDO*

MENDO is an online e-learning system that enables anyone to learn programming, study algorithms, and to solve algorithmic tasks. The system is primarily intended for primary and secondary school students, but it has also been actively used by both coaches and universities for conducting online courses and exercises. It offers automated grading – where solutions, which are submitted as source code, are first compiled, and are then fed batches of input data. At the end, their output is tested for correctness and the solutions are awarded an appropriate number of points. MENDO can work with solutions written in several programming languages (and can easily be extended to support more), can work with various types of tasks (including interactive ones), and the system can restrict programs to arbitrary time and memory limitations. MENDO also supports several different ways of comparing the output and grading the solutions – for example, checking for exact equality, comparing number values, ignoring whitespace, writing custom comparators which can (even) award a custom percentage of the points, etc. Aside from learning from interactive tutorials and solving tasks at any time in the day, MENDO acts as a form of gateway for algorithmic programming, offering a news page, forum, wiki and links to books and guides.

The system itself is also used by the Computer Society of Macedonia for conducting both school, regional and national competitions in informatics, as well as organizing the Macedonian Olympiad in Informatics. Through the years, these competitions were organized in different formats (both online and on-site), with different methods of scoring contestants and offering feedback – requiring constant updates and modifications. The organization of all of these events by CSM has resulted in the creation of most of the tasks that are currently available on the system.

Since the organization of the first event in 2010, the system has had more than 16000+ registered users, 1000+ tasks related to algorithmic programming, 530000+ submitted and graded solutions, and more than 40 tutorials for both starting with programming (specifically, learning C++) and getting familiar with algorithms and data structures. The system supports two languages (Macedonian and English).

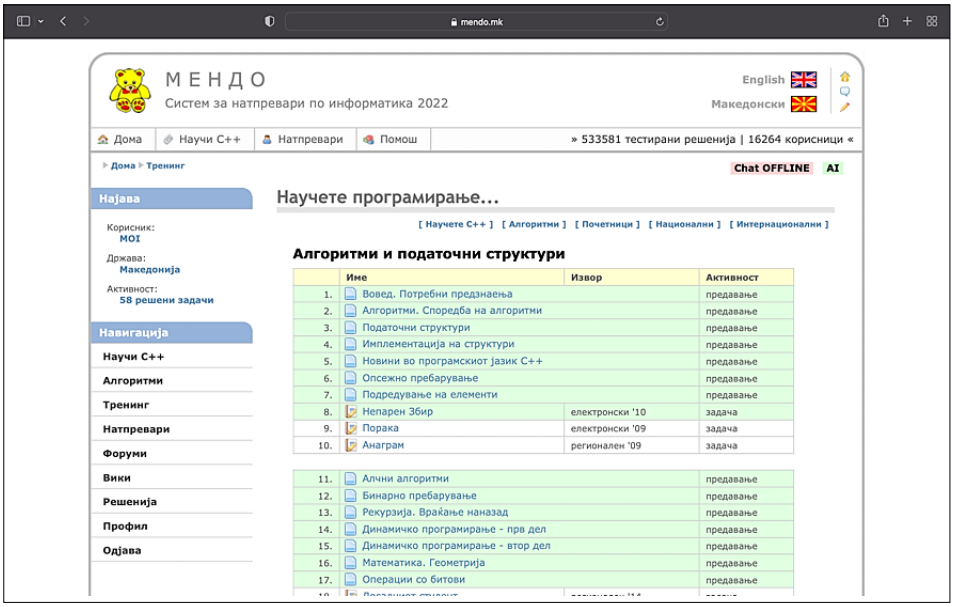


Fig. 1. Screenshot of the MENDO training system – list of tutorials.

The first version of MENDO is described in (Kostadinov *et al.*, 2010), and features several modules: sandbox, grader, controller and auxiliary. Most of the modules were written in Java, and the web application was running on the Apache Tomcat server. It was one of the rare e-learning systems which automatically graded programming solutions on Microsoft Windows. This was done through the sandbox module, which used P/Invoke (Platform Invoke) signatures and Win32 functions to create processes (like the program that a student submits) and group them in jobs. Jobs have the ability of limiting the privileges and resources available to the processes.

In addition, MENDO also controlled other software on the operating system on which it ran, and provided automatic backups for itself and other applications. MySQL was used as an open-source relational database management system to store (most of) the data – like users, task details, submissions, news, etc. The key details that allowed MENDO to support so many features was its modular design which allowed new actions and features to be added by simply implementing appropriate interfaces, as well as the number of quality tests and superb test coverage.

After the release of the system, several updates were made throughout the years, including making design modifications, adding additional features (such as interactive lessons, automated help and hints, or competition statistics – as shown in Fig. 2 below – among others). Some examples of the automated hints include the ability of the system to quickly analyze the source code and, based on the grading results and the source code, to provide feedback to the user regarding simple mistakes such as printing extra data (for example, “The result is: …”), submitting to a wrong task, writing to files, having uninitialized variables, using forbidden imports or functions, etc. Due to the modular nature of the system, and the various test suites created with the first version, most of these

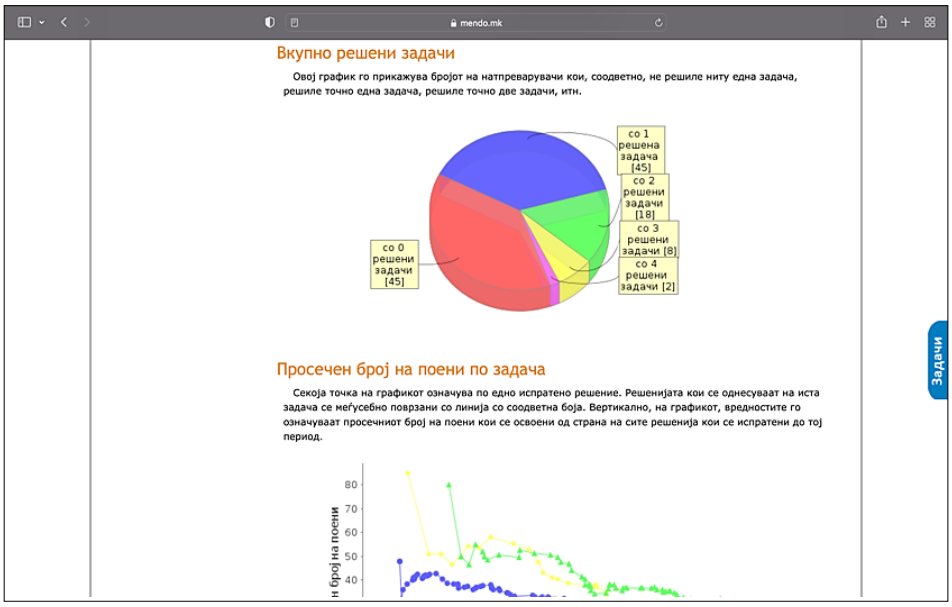


Fig. 2. Screenshot of the MENDO training system – competition statistics.

updates were done smoothly and without any major issues. Because several schools in Macedonia are using browsers which are significantly outdated and don't support many recent features and technologies, we have kept the user interface dependencies fairly consistent. Using virtual machines, we are testing new releases before pushing them to production. Release plans take into consideration firewalls and privacy concerns, and the system itself is visible through a single IP address and doesn't use CDNs.

One of the most significant changes to MENDO, was in regards to where the system is hosted. The first version of the system was running on Microsoft Windows, and all submissions were executed and graded there. In contrast, the latest version of the system is running on Linux, which required a different sandbox implementation and various changes to the connections between MENDO and the other systems that it interfaces with (specifically, the forums and the wiki – as their database connectors had certain issues when we moved them to Linux).

Because the sandbox is just another module in the MENDO system, replacing it was a matter of writing another module that can interact with MENDO and that implements the correct API. We selected around 1000+ submissions from the old system, which had various different verdicts, and tested them with the new sandbox to make sure that the results would be the same – indicating everything works as expected. Only minor issues were detected, like some submissions passing a handful of extra test cases because of the speed difference between the original machine where MENDO was running and the new machine where tests were being executed.

Due to automation tests, all of the requirements were tested, and all discovered issues were solved within days – after which we moved all traffic to the new system when it was confirmed that everything was running smoothly.

3.2. *Bebras*

Bebras is an international challenge organized once per year, which aims to promote informatics and computational thinking among primary and secondary school students. In Macedonia, as in most other countries where the challenge takes place, students participate in the annual Bebras challenge from their school, supervised by their teachers (note: in 2021, because of COVID-19, the challenge took place online).

Countries use different systems to organize the challenge – which could be Moodle, a custom contest management system created specifically for the challenge, pen & paper, or something else. In Macedonia, we use a locally developed system which supports various types of tasks (fill-in-the-blank, multiple-choice questions, and interactive challenges), requires minimum resources, works with different types of browsers and operating systems, and one which supports multiple languages. The system also enables organizers to schedule practice and e-learning sessions, without organizing a specific competition. Because of all the specific requirements around organization, communication, task types, scalability, and performance, the Macedonian Bebras system was developed as a separate system that can function independently and which offers different views for teachers, students and organizers. An example screenshot from the system (student view) is shown on Fig. 3.

The three main concerns we had with running an event with tens of thousands of participants were related to 1) communication, 2) the required performance of the system, and 3) the sizeable number of outdated browsers. To successfully tackle these concerns, the system must be simple to use (so that teachers or students can easily participate with-

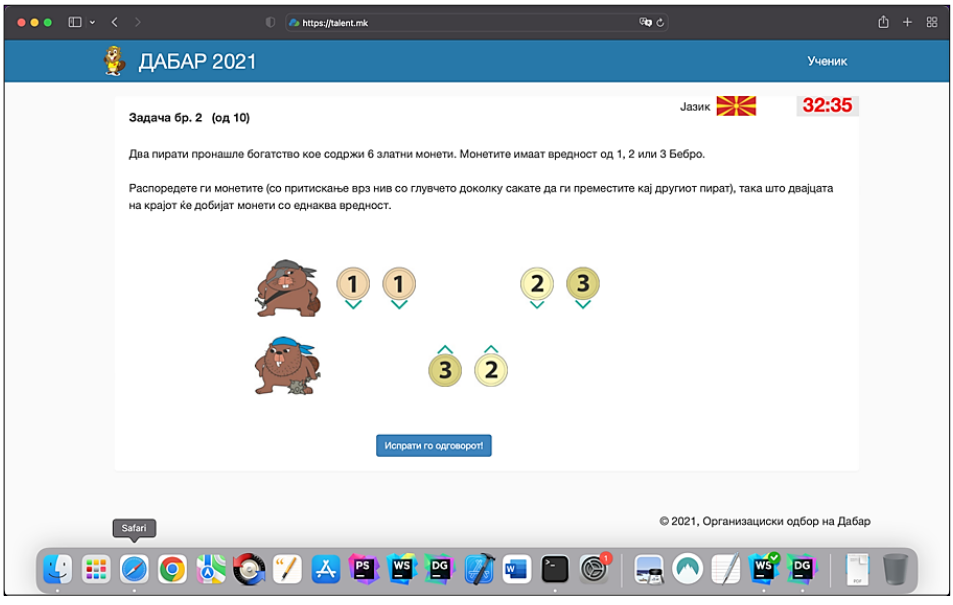


Fig. 3. Screenshot of the Macedonian Bebras system.

out much issues and direct communication with organizers), should be performant so it can support many concurrent users, and the system must support (and be tested) on various browsers. In Macedonia, for example, some computers are limited to using the now (extremely) outdated Firefox 3 (first released in 2008), because of a project organized around that time by the government, aimed at purchasing and delivering new computers to schools. To organize a successful event, plans include testing the system with virtual machines which can run outdated operating systems and browsers, creating benchmarks for stress testing the system, and creating an architecture and content which can be easily extended, tested, and maintained (this is because Bebras usually contains a lot of interactive tasks, which must be properly coded and tested on several browser environments). Cypress was used for testing the final version of the system, before starting the event each year, by automating the flows that students would take on the system and randomly entering the possible answers.

The created system uses an ORM for storing data in two databases – the data is written immediately in one database, while the second one (asynchronously) gets and writes the same data to a ledger in order to make sure all teacher and student information is safely stored – this is used, for example, after the event in order to distribute results and certificates to participants.

Besides structuring the source code properly so it can be extended and modified easily during multiple iterations of the event (for which we followed guidelines such as those outlined in Section 2); having automated backups, proper planning, extensive testing and strong cooperation with teachers was crucial in creating a useful competition management system, and organizing an event with such a large number of participating schools, teachers and students.

3.3. *New E-learning System Based on Item Response Theory*

In (Kostadinov & Stojmenovska, 2022), we describe a distributed e-learning system based on item response theory, that adapts to students, and offers them test questions and learning materials which correspond to their level of knowledge. The system is composed of a main server component where the main data is located, and several separate web agents which should be run at schools (optimally) or other accessible web servers – these agents are the web locations where students connect to. Agents synchronize data with the main server component at configurable regular intervals.

The system has a collection of tasks and explanations, and functions in such a way that it adapts to students based on their determined level of knowledge and skills. It is originally designed to be used by primary and secondary school students, but it might be extended with more content in the future to support more groups and use cases. After a student answers a certain question, the system will show them the correct solution and will also include a brief tutorial regarding the idea and theory behind it. Afterwards, if the student solved the task correctly, the system would generally proceed with a harder task – and vice versa, if the student answers incorrectly, the system will continue with a simpler task. This process is repeated continuously multiple times,

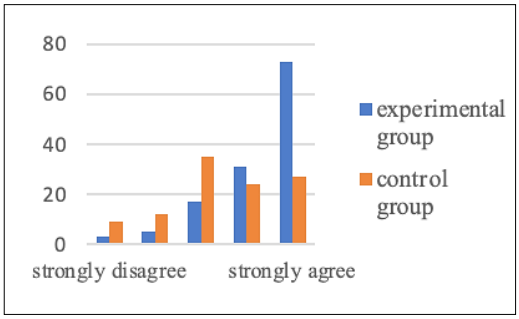


Fig. 4. Students' willingness to use the web applications.

with different tasks, while a student is studying. A more detailed description of the system, and how it uses item response theory to choose the appropriate tasks, can be found in (Kostadinov & Stojmenovska, 2022).

We tested the system with many students (in mathematics), who were split in two categories – one category was using the system as intended, while the other one received questions at random and had access to learning materials in pdf format (commonly shared that way by some schools in our country) instead of the prepared explanations. The gathered results were extremely positive, indicating strong support by students for wanting to continue to use the system.

Because this is a distributed e-learning system, and there are various privacy rules related to storing student data, the web agents that are run at schools are the only ones that store student information. The data synchronized with the main server component is anonymized and is mostly related to task data, answers and content, as defined in the referenced research paper. To guarantee that schools with different machines and operating systems can run the e-learning system on their infrastructure, we use Docker containers and an SQLite relational database for each agent. A well-defined, versioned API is used for communication between the components, which guarantees that future updates will not break existing systems. This is very important when designing distributed systems that are expected to be modified in the future, and especially ones which need to be installed in different locations and environments. (Note that, for the gathered data presented above, the web agents were running on our infrastructure, instead of in schools – this was done in order to simulate a real working environment where we can test the newly created e-learning system, without asking teachers to setup web servers and networks themselves, as they might not have the necessary time, privileges or skills.)

4. Conclusion

Education is a powerful agent of change, and there is a very strong link between education and quality of life. A good education enables people to prepare for the job market, empowers them to understand essential facts about the world around them, and drives

both short- and long-term economic growth. Research has indicated that countries with higher education and literacy rates, also tend to have a stronger GDP and lower unemployment rate (Appiah, 2017). Establishments and governments throughout the world have recognized education as an important tool to transform their organization (on a smaller scale) or society (on a larger scale) to be more productive, stable, independent, and to promote equality among different groups.

The use of educational technology and e-learning is one of the best methods that allows pupils to have similar access to learning opportunities in life. There are different types of e-learning systems, but some of the advantages that are common between them include the ability to learn at one's own pace, to cover material at a time that suits people, and to digest up-to-date content.

Extensibility is one of the main software engineering principles, and it corresponds to enabling, designing, and planning for future growth and updates. There are various strategies that might be employed to extend the life of a software system, which might include wrapping with middleware, renovating/updating, outsourcing, or replacing with another system.

Designing and writing quality source code that follows design principles is also very important, and the same applies to testing, project management and documentation. E-learning systems are sometimes installed on schools' hardware, and these systems might need to follow certain strict privacy guidelines, or to be updated less frequently due to licensing restrictions. Similarly, the school or organization using the system might have network firewalls and/or outdated operating systems and browsers. All of this must be taken into consideration when creating such systems, as it influences the number of options that can be utilized, the required support of multiple API versions, the use of tech, and the ability to outsource services like hosting static files or using Content Delivery Networks (CDNs).

References

Appiah, E.N. (2017). The effect of education expenditure on per capita GDP in developing countries. *International Journal of Economics and Finance*, 9(10), 136–144.

AWS S3 – Amazon Web Services – Cloud Object Storage. (accessed 13/5/2022): `https://aws.amazon.com/s3/`

Janjua, P.Z., Kamal, U.A. (2011). The role of education and income in poverty alleviation: A cross-country analysis. *The Lahore Journal of Economics*, 16(1), 143–172.

Kostadinov, B., Jovanov, M., Stankov, E. (2010). A new design of a system for contest management and grading in informatics competitions. In: *ICT Innovations Conference 2010*, *Web Proceedings*, 87–96.

Kostadinov, B., Stojmenovska I. (2022). Creating a distributed, privacy-aware e-learning system based on item response theory. In: *19th International Conference on Informatics and Information Technologies (CIIT 2022)*.

Lavelle, G. (2005). Practical strategies for dealing with legacy systems. *Risk Management*, 52(2), 45–46.

MENDO – The Macedonian contest management system. (accessed 12/5/2022): `https://mendo.mk/`

Moodle – Open-source learning platform. (accessed 11/5/2022). `https://moodle.org/`

Yagel, R. (2017). Extending Software Systems While Keeping Conceptual Integrity. *In: SEKE* (pp. 432–435).

B. Kostadinov is the founder of Cloud Solutions, an author, and a former competitive programmer. In 2014, he defended his MSc thesis in Intelligent information systems at the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, in Skopje. He is currently a PhD student at UACS, and is one of the organizers of the national competitions in informatics in Macedonia.

I. Stojmenovska is a Dean of the School of Computer Science and Information Technology at the University American College – Skopje. She is a PhD holder in Theoretical Mathematics. Her main research interests are in abstract algebra, in particular including vector valued algebraic structures and combinatorial theory within. She also has research interests related to mathematical education and information systems and management. Prof. Stojmenovska is a member of the European Mathematical Society and appears as an author/coauthor of 40 peer-reviewed research papers published so far.

Olympiads in Informatics, 2022, Vol. 16, 35–42
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.04

What is the Competitive Programming Curriculum?

Antti LAAKSONEN
Department of Computer Science, University of Helsinki
e-mail: ahslaaks@cs.helsinki.fi

Abstract. Competitive programmers learn algorithms and data structures that belong to university computer science curricula. In this paper we go through the "Algorithms and Complexity" knowledge area in the ACM/IEEE curriculum guidelines and determine which of the topics can be learned through competitive programming. After that, we discuss in detail some topics that are different in competitive programming and university courses.

Keywords: data structures, algorithms, curricula, programming contests.

1. Introduction

Solving competitive programming problems can teach many important algorithms and data structures discussed in university algorithms courses. However, there are also differences: some topics are usually only covered either in competitive programming or university courses, but not in both of them. For example, segment trees are used primarily in competitive programming, while Fibonacci heaps are typically only seen in university courses.

There are no clear lists of topics that appear in competitive programming or university courses. The IOI Syllabus (2020) roughly specifies the topics that can be expected in the International Olympiad in Informatics, but there are many topics that are not included in the IOI Syllabus but still appear in other contests, such as the International Collegiate Programming Contest or Google Code Jam. The ACM/IEEE curriculum guidelines (2013) suggest topics that should be included in computer science curricula in universities.

This paper consists of two parts. In the first part, we go through the topics of the "Algorithms and Complexity" knowledge area in the ACM/IEEE curriculum guidelines and determine which topics can be learned through competitive programming. In the second part, we discuss in detail some differences between competitive programming and university textbook topics.

2. Comparison to ACM/IEEE Curriculum Guidelines

The purpose of the ACM/IEEE curriculum guidelines is to recommend topics that should be included in university computer science curricula. The topics have been divided into three categories as follows:

- Core-Tier1 topics are the most fundamental topics and a computer science curriculum should cover them all.
- Core-Tier2 topics are also important and a computer science curriculum should cover all or almost all of them.
- Elective topics are more advanced topics, and a computer science curriculum should also cover many of them.

In this section we go through the topics of the "Algorithms and Complexity" knowledge area in the guidelines. For each topic, we determine if it can be typically learned through competitive programming, i.e., by learning techniques that are needed in programming contests.

2.1. *Basic Analysis*

The challenge in most competitive programming problems is to create efficient algorithms. While competitive programmers routinely work with time complexities and use the Big O notation, not all topics in this group are covered in typical competitive programming training.

Interestingly, the formal definition of the Big O notation has been included in the Core-Tier1 category in the guidelines, while its use belongs to Core-Tier2. However, in competitive programming, it is essential to be able to use the Big O notation when designing algorithms, but it is not necessary to know its formal definition.

Category	Topic	In Contests?
Core-Tier1	Differences among best, expected, and worst case behaviors of an algorithm	Yes
Core-Tier1	Asymptotic analysis of upper and expected complexity bounds	Yes
Core-Tier1	Big O notation: formal definition	No
Core-Tier1	Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential	Yes
Core-Tier1	Empirical measurements of performance	Yes
Core-Tier1	Time and space trade-offs in algorithms	Yes
Core-Tier2	Big O notation: use	Yes
Core-Tier2	Little o, big omega and big theta notation	No
Core-Tier2	Recurrence relations	Yes
Core-Tier2	Analysis of iterative and recursive algorithms	Yes
Core-Tier2	Some version of a Master Theorem	No

2.2. *Algorithmic Strategies*

Category	Topic	In Contests?
Core-Tier1	Brute-force algorithms	Yes
Core-Tier1	Greedy algorithms	Yes
Core-Tier1	Divide-and-conquer	Yes
Core-Tier1	Recursive backtracking	Yes
Core-Tier1	Dynamic programming	Yes
Core-Tier2	Branch-and-bound	No
Core-Tier2	Heuristics	Yes
Core-Tier2	Reduction: transform-and-conquer	Yes

Most of the topics in this group belong to fundamental competitive programming skills. The only exception is the branch-and-bound technique which can be regarded as an advanced technique rarely seen in programming contests. The branch-and-bound technique is used to optimize exhaustive search algorithms, while most competitive programming problems deal with polynomial algorithms. In programming contests, it is often possible to get partial points by implementing a brute force algorithm, but it is not needed to optimize the algorithm.

2.3. *Fundamental Data Structures and Algorithms*

In competitive programming it is important to know how to use efficient algorithms and data structures available in the standard library of the used programming language.

Category	Topic	In Contests?
Core-Tier1	Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max, and mode in a list, approximating the square root of a number, or finding the greatest common divisor	Yes
Core-Tier1	Sequential and binary search algorithms	Yes
Core-Tier1	Worst case quadratic sorting algorithms (selection, insertion)	Yes
Core-Tier1	Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort)	Partially
Core-Tier1	Hash tables, including strategies for avoiding and resolving collisions	Partially
Core-Tier1	Binary search trees • Common operations on binary search trees such as select min, max, insert, delete, iterate over tree	Partially
Core-Tier1	Graphs and graph algorithms • Representations of graphs (e.g., adjacency list, adjacency matrix) • Depth- and breadth-first traversals	Yes
Core-Tier2	Heaps	Partially
Core-Tier2	Graphs and graph algorithms • Shortest-path algorithms (Dijkstra's and Floyd's algorithms) • Minimum spanning tree (Prim's and Kruskal's algorithms)	Yes
Core-Tier2	Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)	Yes

For example, in C++, the function `sort` can be used to efficiently sort an array, the class `unordered_map` implements a hash table, and the class `priority_queue` implements a heap. However, it is not necessary to know how these algorithms and data structures actually work.

This is a fundamental difference between competitive programming and university courses: you can be a successful competitive programmer without knowing, for example, the quicksort algorithm, which is a basic algorithm taught in almost any introductory university course. Such knowledge is not needed in competitive programming because you can use the standard library which provides efficient sorting algorithms. However, you should know how to implement algorithms and data structures that are *not* in the standard library, such as the union-find data structure needed for Kruskal's algorithm.

Note that Prim's and Kruskal's algorithms accomplish the same task, and most competitive programmers seem to prefer Kruskal's algorithm which can be extended to some more advanced problems. While it is interesting to know two different algorithms for determining minimum spanning trees, there is not much use for Prim's algorithm in programming contests.

2.4. *Basic Automata Computability and Complexity*

The topics of this group are outside the scope of competitive programming.

2.5. *Advanced Computational Complexity*

The topics of this group are outside the scope of competitive programming.

2.6. *Advanced Automata Theory and Computability*

The topics of this group are outside the scope of competitive programming.

2.7. *Advanced Data Structures Algorithms and Analysis*

This group has both topics that are regarded as basic topics in competitive programming, such as topological sorting, and advanced topics that are only needed in some difficult problems, such as linear programming.

Many balanced trees, such as AVL trees and red-black trees, are difficult to implement, and in many cases one can just use standard library implementations, such as the classes `map` and `set` in C++. However, balanced trees may be needed in some advanced competitive programming problems where it is required to, for example, split and merge arrays. One popular way to solve such problems is to use the treap data structure whose implementation is relatively easy (if you know a good way to implement it).

Category	Topic	In Contests?
Elective	Balanced trees (e.g., AVL trees, red-black trees, splay trees, treaps)	Yes
Elective	Graphs (e.g., topological sort, finding strongly connected components, matching)	Yes
Elective	Advanced data structures (e.g., B-trees, Fibonacci heaps)	No
Elective	String-based data structures and algorithms (e.g., suffix arrays, suffix trees, tries)	Yes
Elective	Network flows (e.g., max flow [Ford-Fulkerson algorithm], max flow – min cut, maximum bipartite matching)	Yes
Elective	Linear Programming (e.g., duality, simplex method, interior point algorithms)	Partially
Elective	Number-theoretic algorithms (e.g., modular arithmetic, primality testing, integer factorization)	Partially
Elective	Geometric algorithms (e.g., points, line segments, polygons. [properties, intersections], finding convex hull, spatial decomposition, collision detection, geometric search/proximity)	Yes
Elective	Randomized algorithms	Yes
Elective	Stochastic algorithms	Yes
Elective	Approximation algorithms	Yes
Elective	Amortized analysis	Yes
Elective	Probabilistic analysis	Yes
Elective	Online algorithms and competitive analysis	No

It is not clear which data structures exactly belong to "advanced data structures". At least the two mentioned data structures, B-trees and Fibonacci heaps, are not necessary in competitive programming because you can use other data structures instead of them.

3. Competitive Programming vs. University Courses

This section discusses in detail some topics that are different in competitive programming and university courses. In general, competitive programmers prefer techniques that are easy to implement, and try to use algorithms and data structures provided in standard libraries of programming languages.

3.1. *Range Queries*

Range query structures play an important role in competitive programming. For example, using a segment tree (see e.g. Laaksonen, 2020, Section 9.2.2) it is possible to maintain an array of n elements and process two types of queries in O(log n) time: (1) modify an array value, (2) find the maximum value in a given range (subarray).

For some reason, simple range query structures, such as segment trees, are rarely discussed outside competitive programming. Instead, other methods are used to solve problems. For example, Cormen *et al.* (2009, Chapter 14) shows how modified red-black trees can be used to create dynamic tree structures. This approach yields O(log n) operations like segment trees, but it would be very difficult to implement red-black trees during a contest.

There are some popular problems that can be solved using range queries, but are usually solved using another method outside competitive programming. For example, a typical way to count the number of inversions in a permutation in O(n log n) time is to use a modified mergesort algorithm (see e.g. Cormen *et al.*, 2009, p. 42). However, the problem can also be solved using a segment tree that allows us to go through the permutation from left to right and efficiently count the number of previous elements that are larger than the current element.

3.2. *Hashing*

Hash tables are often used in competitive programming as standard library data structures. For example, the C++ classes `unordered_map` and `unordered_set` are based on hash tables. While it is not necessary to implement hash tables and resolve collisions, it is important to understand that hash table data structures may be slow on some inputs.

Some contest systems, such as Codeforces, allow users to send additional inputs (called "hacks") to contest problems. If a solution is based on hashing, it may be possible to hack it by constructing an input where a large number of elements is assigned the same hash value. While hash table operations usually take O(1) time, in this case they can take O(n) time. For example, it is possible to construct inputs where the C++ classes `unordered_map` and `unordered_set` are too slow if they are used in a typical way. We have observed that many students assume that hash tables are always efficient in practice, and it is instructive to see that there are indeed inputs where they are not efficient.

Another topic where hashing is used in competitive programming are string algorithms. Like in the Karp-Rabin pattern matching algorithm (1989), we can compare substrings of strings in O(1) time after preprocessing the strings using the hash values of the substrings (see e.g. Laaksonen, 2020, Section 14.2). A speciality in competitive programming is that it is often assumed that there are no collisions, i.e., if two substrings have the same hash value, they also have the same content. When hashing is properly implemented (Pachocki and Radoszewski, 2013), many string problems can be solved using the technique.

3.3. *Binary Search*

A traditional way to use binary search is to efficiently search for values in a sorted array in O(log n) time. In competitive programming binary search is not often used in that way, because we can either use a standard library implementation of binary search (such as the `lower_bound` and `upper_bound` functions in C++) or we can use an efficient data structure that is available in the standard library.

Instead, binary search is often used as an algorithm design technique: when we know that a function $f(x)$ has value 0 when $x < k$ and value 1 when $x \geq k$, we can efficiently

find the smallest x value such that $f(x) = 1$ using binary search (see e.g. Laaksonen, 2020, Section 4.3.2). Surprisingly, this way to use binary search is not often discussed in algorithms textbooks. In some cases the reason may be that there is another way to solve the problem without using binary search.

3.4. *Dijkstra's Algorithm*

The usual way to implement Dijkstra's algorithm in textbooks (see e.g. Cormen *et al.*, 2009, Section 24.3) is to build a heap that contains a distance to each node of the graph. Initially each distance is infinite, and the distances are updated during the algorithm using the decrease-key heap operation. This implementation works in $O(m \log n)$ time where n and m represent the number of nodes and edges, assuming each element is reachable from the starting node.

The problem in such an implementation is that heap implementations in standard libraries (such as the `priority_queue` class in C++) typically don't support the decrease-key operation. For this reason, it would be necessary to implement a custom heap instead of using the standard library data structure. However, in competitive programming, another version of Dijkstra's algorithm is used (see e.g. Laaksonen, 2020, Section 7.3.2) which doesn't update the distances in the heap but instead adds a new distance to the heap when a distance changes. This allows us to use a standard library heap implementation in the algorithm. It can be shown that this version of Dijkstra's algorithm also works in $O(m \log n)$ time even if the number of elements in the heap may be larger than in the textbook implementation.

This is an example of a tendency that can be seen in competitive programming: new ways are invented to use standard library algorithms and data structures whenever possible, which makes implementations shorter and saves time during contests.

3.5. *Divide-and-conquer*

The divide-and-conquer technique is often regarded as a basic algorithm design technique. For example, Kleinberg and Tardos (2006) devote an entire chapter to the technique, and discuss algorithms such as mergesort, finding the closest pair of points and the FFT algorithm. However, using the divide-and-conquer technique is rarely required in competitive programming problems.

It seems that the divide-and-conquer technique is often used indirectly in competitive programming problems. While mergesort is an important algorithm, it is not necessary to implement it because we can use the standard library implementation, such as the `sort` function in C++, which may use mergesort. The FFT algorithm is required in some advanced competitive programming problems, but it is often used as a prewritten black box algorithm.

There is another way to solve the closest pair of points problem that differs from the traditional divide-and-conquer algorithm which divides the points into two sets and

recursively solves the problem for each group and then combines the results. Instead, we can process the points from left to right and use a balanced binary tree to maintain a set of relevant points (see e.g. Laaksonen, 2020, Section 13.2.2). In this implementation, we can think that the divide-and-conquer idea is hidden in the balanced binary tree and it can't be seen in the main algorithm.

4. Conclusion

Competitive programming covers many, but not all, of the algorithm design and data structures topics in the "Algorithms and Complexity" knowledge area in the ACM/IEEE curriculum guidelines. There are also topics that are different in competitive programming and university courses, and there are competitive programming approaches that rarely appear in textbooks.

Some theoretical topics are only rarely needed in competitive programming. However, an interesting question for future work is what competitive programmers really know besides the topics relevant in programming contests. For example, do they usually know how the heap data structure works, even if it is not necessary to implement a heap during a contest?

References

ACM/IEEE (2013). Curriculum Guidelines for Undergraduate Programs in Computer Science. Available online at: `https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf`

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2009). *Introduction to Algorithms* (Third Edition). MIT Press.

IOI Syllabus (2020). Available online at: `https://ioinformatics.org/files/ioi-syllabus-2020.pdf`

Karp, R.M., Rabin, M.O. (1989). Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2), 249–260.

Kleinberg, J., Tardos, E. (2006). *Algorithm Design*. Addison–Wesley.

Laaksonen, A. (2017). A Competitive programming approach to a University introductory algorithms course. *Olympiads in Informatics*, 11, 87–92.

Laaksonen, A. (2020). *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests* (Second Edition). Springer.

Pachocki, J., Radoszewski, J. (2013). Where to use and how not to use polynomial string hashing. *Olympiads in Informatics*, 7, 90–100.

A. Laaksonen works as a university lecturer at the Department of Computer Science of the University of Helsinki. He is one of the organizers of the Finnish Olympiad in Informatics and has written a book on competitive programming. He is also a developer of the CSES online judge.

Olympiads in Informatics, 2022, Vol. 16, 43–53
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.05

Initial Learning of Textual Programming at School: Evolution of Outreach Activities

Martina LANDMAN, Gerald FUTSCHEK,
Svetlana UNKOVIC, Florentina VOBORIL

TU Wien, Institute of Information Systems Engineering
Favoritenstraße 9-11, 1040 Vienna, Austria
e-mail: martina.landman@tuwien.ac.at, gerald.futschek@tuwien.ac.at,
svetlana.unkovic@tuwien.ac.at, florentina.voboril@tuwien.ac.at

Abstract. Learning programming is at least to some extent part of school curricula in nearly all countries. But in practice there is still a shortage of teachers that got a relevant education in computer science and, in particular, computer programming in almost all countries. Another challenge of teaching programming in schools is the heterogeneity in prior education and programming knowledge of students. This ranges from no prior knowledge to deep experience. We show in this paper how we outreach from our university to teachers and school classes using learning materials and personal support. We also show three stages in the development of our outreach activities and argue why we needed to evolve to the next stages.

Keywords: learning programming, outreach to school, MOOC, processing, workshops.

1. Introduction

Computer programming is not new at school and is part of school curricula in most countries at least to some extent (Bocconi *et al*., 2022). The use of text-based programming languages is part of the curriculum, especially at the upper secondary level. There exists already a wide variety of online materials and programming courses that are suitable for use at school.

But in practice there are several reasons why in many school classes a satisfiable programming education cannot be given:

1. Engagement of teachers without informatics education due to lack of informatics teachers.
2. Further education of in-service teachers of other subjects is often hard to achieve.
3. The programming environments for textual programming are often overly complex.

4. Teaching programming needs a lot of effort: posing of appropriate tasks, assessing of student programs, reacting to students' problems in writing computer programs.
5. Need of tasks for students of different programming skills levels.

Lack of educated informatics teachers is a major problem in informatics education at school. A major challenge is to educate enough informatics teachers, see (Webb *et al.*, 2017). Most of the countries allow non-informatics teachers to teach programming and other computer science topics. These teachers are usually not experienced in computer programming. They need additional support.

Therefore, our requirements for teaching support in the field of textual computer programming are:

1. Use of a programming language that is easy to learn.
2. Provision of teaching and learning materials: videos, tasks, explanations, tests.
3. Personal support for teachers.

So, our approach is to support teachers in teaching initial programming by personal support and by adaptable teaching material. We describe in this paper three stages of concepts to convey our concept and materials to school classes. We started with use of a MOOC, moved on to a flexible adaptable online programming course and finally provided introductory workshops at school classes.

2. A Programming MOOC for First Year University Students as Basis

When we decided in 2020 to outreach to schools with a textual programming activity, we had already developed in 2018 a MOOC that was based on the programming language Processing, a variant of Java with high potential of learning textual programming. The primary target group of this MOOC was beginners at university level. This MOOC was successfully used in pre-university courses to prepare potential students for MINT studies. A detailed description of concept and structure of this MOOC can be found in Wetzinger *et al*. (2018). Although we implemented some cooperative tasks, the typical use of this MOOC was in self-study during summertime by the already enrolled students.

During construction of this MOOC, we had already a potential use at school level in mind and therefore expected from the users only such pre-knowledge, which is standard pre-knowledge in upper secondary schools. The MOOC consists of two parts and is still available at the public MOOC platform iMooX (iMooX Processing course). All the ten chapters contain several programming tasks, for all of them are also solution programs available.

We advertised this course from the very beginning also for use in school classes, but the use was limited to very experienced teachers that were able to integrate this educational resource into their teaching praxis.

We observed the following barriers for a wider use:

- The teachers cannot observe the learning activities of their students on the iMooX platform, they therefore used an additional platform or asked for copies of the course.

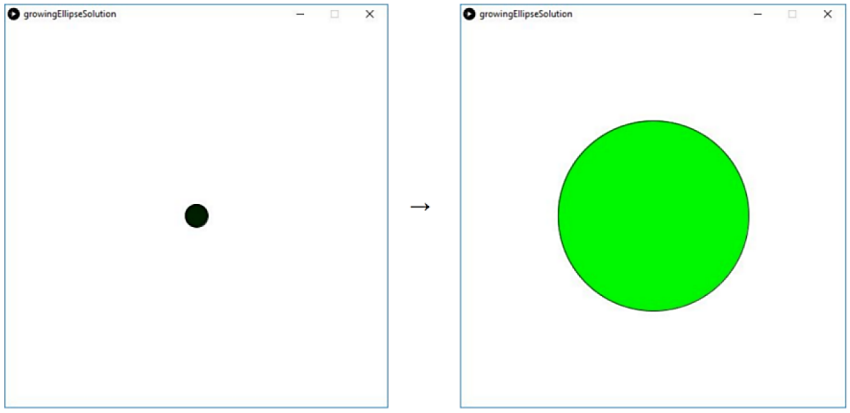
- All provided tasks had already solutions, there was a need for additional tasks for homework and tests.

To demonstrate the transition of our activities we show how a selected task changed from phase to phase.

We give an example of an exercise of the MOOC involving graphics and simple animation, where students must adapt a given code:

The goal of this assignment is to introduce first year university students to the use of control variables and the possibility to bound their value with the modulo operator. This kind of task is intended for interested pre-university students and does not involve hints for a possible solution.

By executing the program (L_Animation_UE2.pde), after your successful adaption, a small black circle should be displayed, which continuously grows to a large green circle. As soon as it has reached its maximum size of 255 it should start again as a small black circle with size 0 and grow again continuously.



Your task now is to add at the marked position (//TODO) an appropriate assignment to the variable *colorIntensity*.

```
int colorIntensity = 0;
void setup(){
  size(500, 500);
}

void draw(){
  background(255);
  fill(0, colorIntensity, 0);
  ellipse(250, 250, colorIntensity, colorIntensity);

  //TODO
}
```

Fig. 1. MOOC-Task including graphics and animations.

3. Online Programming Course for School Classes

We went one step further and wanted to reach a larger and younger target audience with our online materials from the MOOC. As already mentioned, the preparation of the content and supervision in the classroom represents an enormous amount of work for teachers in schools. That is why the idea came up to adapt the existing MOOC contents to students and support teachers with and without programming experience in their computer science lessons. A student-friendly Moodle course for school classes from the 9th grade was created, which is divided into the following fourteen teaching units:

1) Introduction & Basic Figures.
2) Variables.
3) Animations.
4) Characters and strings.
5) Truth values and colors.
6) Logical operators.
7) Branches.
8) While-loops.
9) For-loops.
10) Subprograms.
11) Troubleshooting.
12) One-dimensional arrays.
13) Two-dimensional arrays.
14) Recursion.

Many in-service teachers are not well trained in teaching programming to their students because that was not part of their education in the past. Due to the massive lack of computer science teachers, schools employ either people who are working in companies outside school and have great programming experience but therefore less pedagogical and didactical knowledge or no-specialist teachers. These non-specialist teachers studied different subjects and additionally took a short course regarding informatics in the past so that they are allowed and able to teach the basics. Unfortunately, many of them show problems in programming because of the missing programming experience. The adaption of this course was intended to help and support those teachers. Another reason why we chose the target group from 9th grade upwards to pre-university school is the mandatory programming part in the school curricula in Austria.

Like the MOOC course, each lesson is dedicated to a programming concept. In contrast to the MOOC for first-year students, the students do not work on a project, but rather work on shorter and simpler exercises with the aim of motivating the learners with small moments of success and developing joy in programming. The content is consolidated with the help of explanatory videos, scripts, cheat sheets, self-examinations and (further) exercises. One of those exercises is stated below.

This exercise is based on the exercise, presented at the previous section, but it was adjusted to the new target group. It uses simpler language and it is divided into two sub

steps. In contrast to the version before, the description of the exercise is embedded in Moodle and the resulting program is shown as animation.

Students also have the opportunity to ask questions in the forum if needed. It must be said that the teachers have access to all tasks and their solutions. Depending on the teaching style, the teacher can make the solutions available to the students or discuss specific programs with them.

Since different motives play a role in taking part in the course, the teachers can basically freely design the teaching units and contents. Some classes, for example, have already gained programming experience and only want to learn a new programming language, and others not at all. This then of course rubs off on the course of the lesson and is crucial to how well you can let the students work independently. Nevertheless, we offer a possible process for teachers, as well as a possible assessment scheme. Our idea of the teaching units corresponds to a flipped classroom concept (Sobral 2021). The learners work on the content via videos, scripts and self-examinations from home and can then work on the programming examples in computer science lessons. The computer science class, we could say, acts as a kind of test laboratory. Additionally, teachers have insight into the processing progress of each learner via the Moodle course. In this

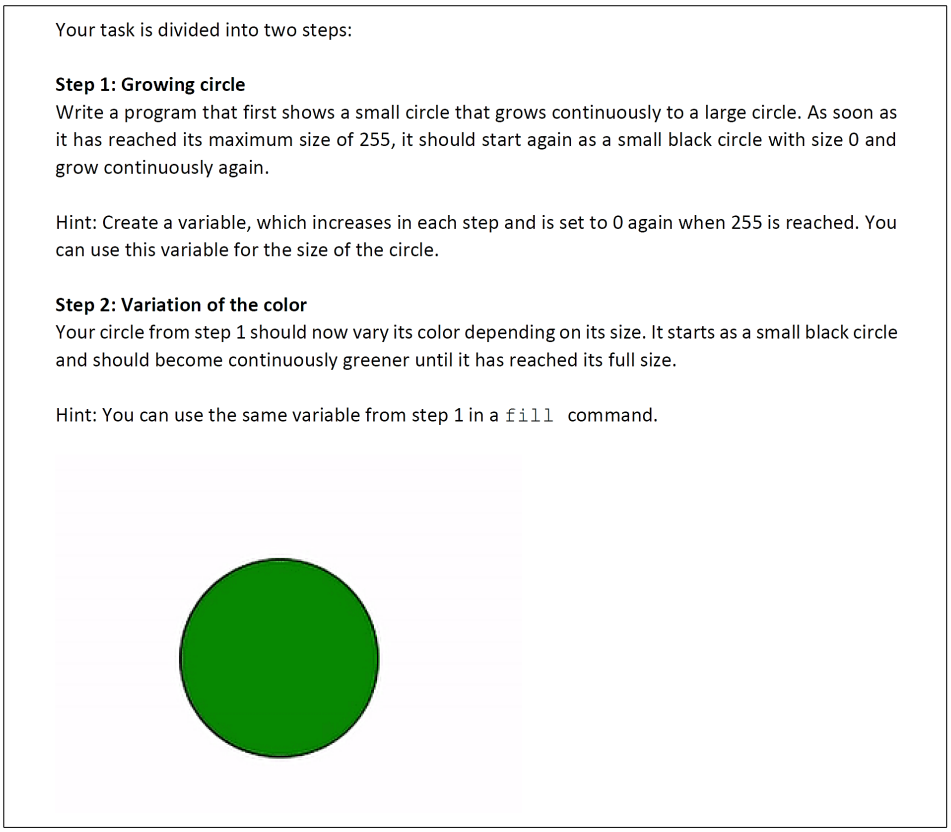


Fig. 2. Online-Course task, which includes animations and hints.

way, the teachers can see the difficulties that the students encounter and this can help to develop a better understanding to their problems. It is also possible to involve employees of the TU Vienna in the school classes. The further the school classes progress with the topics, the more difficult the exercises can become. It may make sense to ask for additional help.

As already mentioned, there are no restrictions on who uses the course. Both teachers with and without programming experience can access the course. From experience, however, most teachers without previous programming experience usually do not trust themselves in teaching the contents of the Moodle course. That is why we offer a slimmed-down version in form of a workshop.

4. From an Online Course to a Workshop in School

After developing and redesigning the MOOC to an Online Course format for School classes, both courses are still available.

- The MOOC, which can be used by private individuals and is accessible via the iMooX platform (registration required). It is split up in two parts.
- The school course with fourteen units, which can be redesigned by the teacher and offers the possibility of student assessment.

The second version, which is more attractive to teachers, is not available through a self-subscription and must be requested through a request form on the same website. Students can also be inserted into this course. However, for this the teacher must send us the students' data so that we can insert them. We chose this format in Moodle to get in touch with every teacher and have the possibility to give them a separate "room" for their teaching with chances for assessment.

In the process, we encountered several difficulties and uncertainties on the part of the teachers. Teachers were unsure about data protection. As the student data is entered into our platform, the parents of the children must agree. There are schools that already cover this case by having the parents sign when the students are admitted to the school. Thus, for some teachers this is not a problem, but for others it is. Therefore, it was decided that the course would be unlocked for teachers to copy. This means that teachers can copy the course and paste it into the school's own Moodle instance.

This leads us to another problem: the transparency of the use of the course. As soon as teachers copy the course into their own platform, we naturally lose the number of students who use the course and who have been reached.

From a survey in autumn 2021, we were able to find out that 93 students were reached. We assume that considerably more students were reached than the survey recorded, as only seven teachers completed the survey, but a total of 17 courses were created. Extrapolating this would give a notional number of 225 students reached. Although 17 new courses were created for teachers in 2021, only one course was created in the first quarter of 2022.

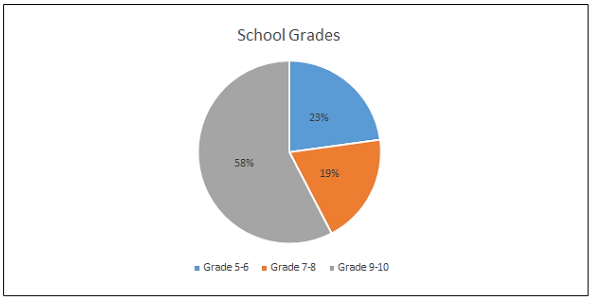


Fig. 3. School Grades in which the online course was used in 2021.

This leads us to the third problem: few interested parties. The decreasing number of interested parties is probably related to the structure of the course, which is difficult to integrate into face-to-face teaching. In the school year 2021/22, no official Distance Learning has been done so far, so it takes a lot of effort for teachers to integrate the content into the classroom. Of course, they are free to use and adapt the content, but many do not have the time.

For these reasons, we have gone one step further and redesigned the course again, consisting of fourteen units, so that you can book a short "Programming with Processing" workshop, taught in 2–4 units, which means 1–2 double lessons in School – 100 Minutes each. For that purpose, we go with our team into schools and do a "Programming with Processing" workshop there in the class. The teacher is assisting our trainers. Afterwards it is up to the teacher if he or she wants to continue with the topic and uses the material from the online course.

From our survey in autumn 2021 we learned, most of the teachers use the course in the 9th to 10th grade. This was the reason, why we designed the workshop for this grade.

5. Programming Workshop in Schools

The following adaptations to the previous version were made:

- From our survey in autumn 2021 we know that the last four units where never done by the teachers in school. Only two teachers did it further than the first two units. So, we decided to make a course with the most essential parts of programming for the first time. The programme was taken (partly) from the first three units of the course. It deals with the most essential contents that are necessary when programming for the first time. The students learn about branching and loops. They learn the structure of a programme.
- The parts that were available as explanatory videos in the original course are now taken over by one of our trainers. This change was made because of the presence workshop format.
- It is taught in a team-teaching format, with students taking turns working together and then doing free exercises.

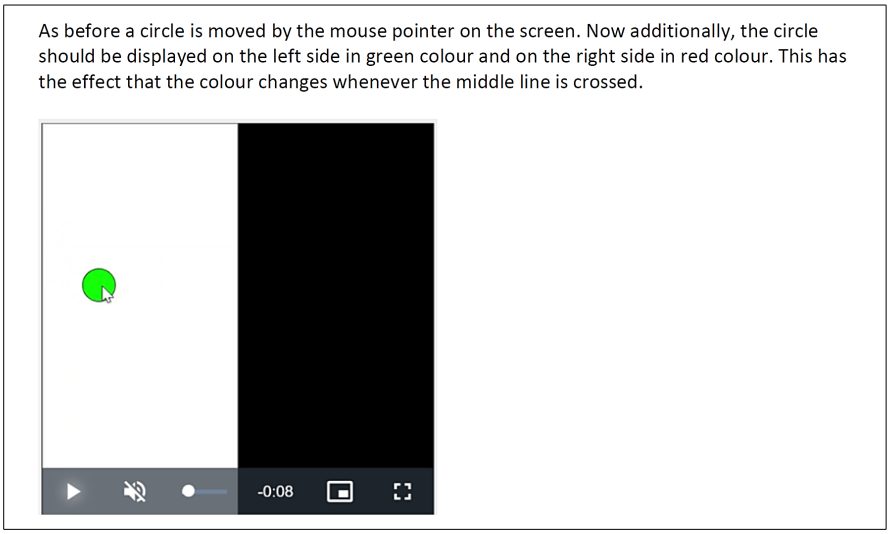


Fig. 4. Task in school workshop, including animations.

- The exercises were designed in such a way that there is always one compulsory task and several additional voluntary exercises. In the pilot test, the weakest students only completed the compulsory task and the most talented completed one or two of the additional tasks.

After a task that allows a circle to follow the mouse pointer on the screen the following task may be given: Fig. 4.

Compared to the exercises before, we included more interactive tasks, which e.g., depend on the position of the mouse pointer. This was especially done to increase the motivation of the students. Also, it fits better to the target group of school students who expect game-like interactions.

The time schedule of the workshop looks as follows:

Part 1
Content
Welcome, introduction and overview
Processing: The programming environment ● Processing surface ● Coordinate system ● Comments ● Difference between setup() and draw()
The first program ● typing of commands together
Explanation of terms: commands, parameters and functions ● Colors (with color selection) ● Importance of the command sequence

Exercises about basic figures
Predefined variables • height, width • mouseX, mouseY
Exercises about predefined variables

Block 2

Content
Repetition of contents from block 1
Use of predefined variables in branches new predefined variables: mousePressed, keyPressed, key
Exercises about simple comparisons
Composite comparisons
Exercises about composite comparisons

We piloted the adapted material in two different 9th grade classes with each one week between workshop part 1 and 2. We had one trainer, one teacher assisting and an additional team member for observing and assisting. The conclusion was:

- After the piloted version we had to adapt the schedule to fit better.
- The exercises that were chosen where on a proficient level: The weakest students were able to fulfil all the mandatory tasks in time. This was a good resume for us. The best students did not all of the voluntary tasks, so we have a bit room upwards, if we have an extremely talented group next time.
- Regarding to the teacher, the class gave more attention to the external trainer and where more quiet than usual. This is another opportunity to teach effectively programming in schools when there are external people and not the usual teacher.
- A problem which we faced in the second part of the workshop, one week after the first part, where students that missed the first part. We installed a buddy-system, where one student helped the other to catch up with the whole group.
- One student was extremely interested in programming. Afterwards the class teacher told us, that this student is not interested at all in the "normal informatics lessons stuff," like using office software.

6. Summary

We showed in this short report how we changed our offer to schoolteachers to support them in teaching initial computer programming. We started with a MOOC for learning Processing consisting of ten chapters, addressing potential novices at University MINT studies. To address students at schools and to support their teachers we adopted this course in a way that we added many tasks and the tasks were made easier understandable. The result was a course with fourteen chapters that fit into usual school lectures.

The last step of adaption was made to support teachers with less experience in teaching programming. We created and offered a workshop of two parts à 100 Minutes that conveys all necessary programming skills to start programming animations. We learned that it is absolutely necessary to understand and fulfill the needs of the target group. Then it is possible to teach programming at various levels of the target group. We are still observing and collecting experiences to improve the course in the future, it is an ongoing process.

References

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M.A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V. and Stupurienė, G. (2022). Reviewing Computational Thinking in Compulsory Education, Inamorato Dos Santos, A., Cachia, R., Giannoutsou, N. and Punie, Y. editor(s), *Publications Office of the European Union*, Luxembourg, ISBN 978-92-76-47208-7 (online), doi:10.2760/126955 (online), JRC128347.

eduLAB online course for school classes: `https://edulab.ifs.tuwien.ac.at/programme/onlinekurs-fuer-schulen/`

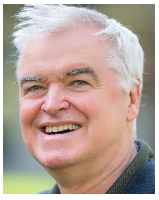
iMooX Processing course: `https://imoox.at/course/processing1`

Sobral R.S. (2021). Flipped classrooms for introductory computer programming courses. *International Journal of Information and Education Technology*, 11(4), 178–183. `http://www.ijiet.org/vol11/1508-AE002.pdf`

Webb, M. *et al.* (2017). computer science in the school curriculum: Issues and challenges. In: Tatnall, A., Webb, M. (eds) *Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing. WCCE 2017. IFIP Advances in Information and Communication Technology, vol 515.* Springer, Cham. `https://doi.org/10.1007/978-3-319-74310-3_43`

Wetzinger, E., Standl, B., Futschek, G. (2018). Developing a MOOC on introductory programming as additional preparation course for CS Freshmen. In: *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), pp. 1663–1672.

M. Landman – Researcher at TU Wien and member of the Informatics eduLAB group in the research unit of Information & Software Engineering since 2021. She is also a teacher and has experience in teaching computer science from 5th to 12th grade. She organizes the computer science faculty's school outreach activities, where she develops, organizes and conducts weekly workshops for school classes.



G. Futschek – is Professor at Institute of Information Systems Engineering and head of TU Wien Informatics eduLAB. His research area in informatics didactics is founded on his computer science background. His mission is to convey computational thinking and concepts of informatics in a way that makes fun and motivates to learn more. He does this in his lectures and with his group by outreaching to schools.

S. Unkovic – is a student assistant at TU Wien Informatics eduLAB since 2020. Currently she is completing her studies in mathematics and computer science. Her main work focuses on the creation and further development of creative materials for students and teachers. These materials are supposed to support them building a better understanding of computer science concepts.

F. Voboril – studies Software and Information Engineering at TU Wien, where she also works as student assistant. During her school years she attended courses in Robotics and participated in some competitions at this field. To share her knowledge, she teaches children in Robotics and Informatics. Since 2019 she is voluntary member in the Austrian team for the Bebras International Challenge on Informatics and Computational Thinking.

Olympiads in Informatics, 2022, Vol. 16, 55–73
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.06

Error Handling in XLogoOnline

Jacqueline STAUB
Department of Computer Science Trier University
email: staub@uni-trier.de

Abstract. This article presents an approach to error handling with Logo novices from first to sixth grade. While structural programming errors can be mostly prevented using visual programming interfaces, logical errors must be dealt with from the very beginning. We have developed a turtle graphic task collection with an integrated solution verification to determine logical correctness of student solutions. Once learners transition from block- to text-based programming, a sizeable field of possible structural errors opens up. Thanks to static program analysis most structural programming errors can be detected while the programmer is still typing. Using a debugger, finally, programmers of all ages have the possibility to observe their program's behavior while stepping through the code. We summarize the error handling tools provided as part of the XLogoOnline programming environment and explain how students can use such aids to attain a constructive attitude towards errors.

Keywords: educational programming, K--6, turtle graphics, Logo, error diagnosis.

1. Introduction

More and more countries have recently started to include computer science (and thus also programming) in their public curricula. With this, even elementary school students now have a chance to learn how to program in various countries around the world. This political change opens up many opportunities, but also raises unresolved questions: What should programming instruction look like in kindergarten when children are not yet literate? How can a teacher provide individualized support to students when programming is known to be an error-prone activity and every child in a class is most likely struggling with errors? We want to clarify these questions by presenting a spiral curriculum for programming in K-6 as well as an approach to error handling. This article summarizes the core ideas presented in XLogoOnline (Staub, 2021; Menta *et al.*, 2019; Forster *et al.*, 2018; Staub *et al.*, 2021).

The recent introduction of computer science in public schools provides children with the opportunity to explore the exciting world of algorithms by learning to program. In order to teach basic programming concepts in an age-appropriate way, both teaching materials (Komm *et al.*, 2020; Hromkovič and Kohn, 2018; Hromkovic, 2010)

and learning environments (Trachsler, 2018; Maloney *et al.*, 2010; Cooper *et al.*, 2000; Hromkovič *et al.*, 2017b; Kohn and Manaris, 2020; Repenning and Ioannidou, 2006) must be readily available. Around the world, there are millions of students that are expected to develop the ability to solve problems algorithmically by the time they enter lower secondary school. Researchers proposed generic frameworks for fostering algorithmic thinking (Dagienė *et al.*, 2021) in computer science as well as concepts with a focus on programming (Hromkovič *et al.*, 2017a; Hromkovič *et al.*, 2016).

Programming is a creative but error prone (Fitzgerald *et al.*, 2008; Gugerty and Olson, 1986) form of learning that enables teachers and students to explore and develop algorithms for a wide range of different problem classes. In essence, programming is a form of communication with a computer; for this a language is required that the computer "understands". Programming languages, much like natural languages, have a vocabulary and a grammar. In contrast to natural languages, however, programming languages are more precise and computers lack the ability to interpret ambiguous statements. As a result, programmers must take special care to express their thoughts accurately. Errors are inevitable and learning to resolve them is a core competence any programmer needs to establish.

The spectrum of programming errors ranges from simple structural errors (e.g., incorrect punctuation, missing or incorrect arguments, and unbalanced parentheses) to more complex logical errors (e.g., reversed loop conditions, forgotten invariants). While both of these error classes are the bread and butter of any programmer regardless of age and experience, there are some error classes that are specific to children. Programming is possible from as young as six or seven years; at that age novices are able to understand basic programming concepts and anticipate program logic (Ettinger, 2012), but they have difficulty expressing themselves in a written language (Solomon, 1993). In order to prevent structural issues due to typing, block-based programming interfaces have been developed and are used in various environments nowadays (Weintrop, 2019).

It is our firm belief that error handling is one of the most essential skills a young programmer needs to acquire by the end of their programming education. Programming environments therefore need to be equipped with error diagnosis tools that are dedicated to the use in programming classes to handle logical errors from the very beginning and later on structural errors in addition. In this work, we summarize the tools we employed in our programming environment XLogoOnline. The environment allows students to begin programming without literacy skills, and provides useful tools for finding, analyzing, and fixing programming errors throughout programming instruction from kindergarten to sixth grade.

Section 2 provides a overview on the linguistic features of the famous programming language Logo and its decade-old traditions yet without diving into the application domain turtle graphics. More detail on our curriculum, the turtle philosophy, and the concrete implementation of these ideas in XLogoOnline follow in Section 3. Finally, Section 4 and Section 5 discuss the concrete details of how error handling is managed in XLogoOnline before concluding.

2. The Logo Programming Language

More than 50 years ago, there first emerged the idea of creating dedicated programming languages that could be used in an educational context with children and adolescents. Although computers were anything but a commodity at that time and the few available models were mostly reserved for universities, thanks to the initiative of Seymour Papert and his team (Papert, 1980; Solomon *et al.*, 2020), school kids as young as secondary or even primary school were able to enjoy the pleasure of programming. Papert and colleagues built the foundation and revolutionized the field of programming education by designing a programming language that specifically targeted to the needs of novices. The resulting language, Logo, is distinguished by its exceptionally minimal and elegant syntax, which (despite its age) still stands out today due to three unique attributes:

- **Whitespace as statement delimiter:** Instead of classical statement delimiters like semicolons or line breaks (as known from Java or Python respectively), Logo allows any number of statements to be placed side by side without an additional delimiter other than a bare space. Three procedure calls `foo`, `bar` and `baz` can thus be simply concatenated without requiring any additional characters in between: `foo bar baz`
- **Whitespace as argument delimiter and no brackets:** Like many other programming languages, Logo supports parameters for procedures. How many parameters a procedure can take depends on its specification; from the linguistic point of view any number of arguments are possible. Moreover, while arguments in other programming languages usually are surrounded by parentheses and require to be separated by a dedicated syntactic character, say a comma, Logo allows a simple and elegant alternative – no parentheses required and arguments are separated by a single white space: `mod 4 2`
- **Deliberate reduction to the minimum:** Cognitive load in programming is reflected (among others) by the number and complexity of the programming concepts used (Hromkovič *et al.*, 2017b). Instead of overburdening students with a multitude of different programming constructs in one go, the Logo philosophy proposes to construct their own more complex language elements. In a truly constructivist manner, the programming language "grows" together with the programmer's proficiency level.

In addition to these purely linguistic aspects, Logo is also famous for its world-renowned application domain Turtle Graphics. The concept of the Turtle as well as a corresponding curriculum for K–6 are presented in the following section.

3. A Spiracl Curriculum for Programming Classes in K-6

Turtle Graphics has stood the test of time and proved a valuable way of introducing beginners to programming. The principle is based on the visualization of the program

execution by means of a virtual or physical computing agent, i.e., the "turtle". This turtle is, in essence, an object whose position and orientation in space can be changed programmatically. Students understand the turtle as a tangible representation of the abstract executions mechanism used in a computer. In order to control its behavior, students need to learn the turtle's "mother tongue" Logo which initially consists of four simple commands:

- **`Forward:`** The turtle moves straight ahead [by a given number of pixels].
- **`Back:`** The turtle moves backwards [by a given number of pixels].
- **`Right:`** The turtle turns to the right [a given angle].
- **`Left:`** The turtle turns to the left [a given angle].

With these four basic commands it is possible to solve simple navigation tasks of the form "guide the turtle from A to B without visiting C along the way" (as illustrated in Fig. 1) to more complex geometric tasks (as shown in Fig. 2). All of these tasks could be posed both in a block-based and text-based interface. One aspect that distinguishes navigation tasks from simple geometry tasks is that for pure navigation in a grid no parameters are required (i.e., unit distances and angles can be used) whereas geometry tasks profit from parameterized basic commands.

Our approach proposes a spiral approach for programming instruction from kindergarten to grade six. Notable milestones in the intended learning progress can be summarized in three stages:

1. **Stage 1 (kindergarten to 2nd grade):** In the youngest age group, children work in a block-based interface with basic commands that do not include parameters (i.e., the `forward` and `back` movement commands cause motion at unit distances while the `right` and `left` rotation commands only perform 90 degree turns). In this framework, learners immerse themselves in the following task types: (i) building sequences of basic commands, (ii) creat-

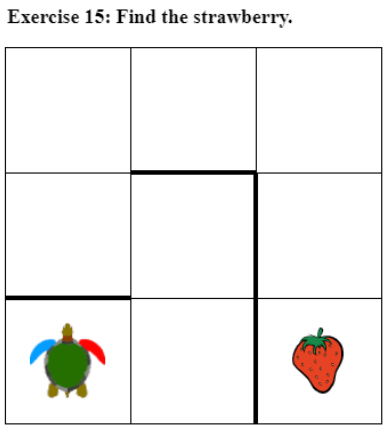


Fig. 1. Navigation task.

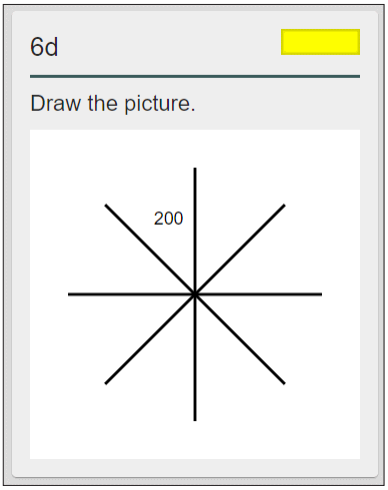


Fig. 2. Geometry task.

ing programs under constraints, (iii) working with colors, (iv) covering longer distances with `repeat`, (v) shortening repetitive program sequences with `repeat`.

2. **Stage 2 (3rd and 4th grade):** After the first stage, students transition from the previous navigation-based tasks to the more traditional geometry tasks. For this purpose, the basic commands `forward`, `back`, `right` and `left` are extended with parameters (i.e. the two movement commands allow to draw lines of arbitrary length and the rotation commands can cause rotations of arbitrary angles), while the interface stays block-based. In terms of content, this second stage focuses on the following concepts: (i) building sequences of basic commands, (ii) creating programs under restrictions, (iii) working with colors, (iv) shortening repetitive program sequences using `repeat`, (v) sequences of `repeat`, (vi) nested `repeat`.
3. **Stage 3 (5th and 6th grade):** Finally, the transition from block-based to text-based programming takes place. While the Turtle Graphics application area remains the same, this step mainly changes the input form and the depth of the concepts covered in the curriculum. Students engage in the following types of tasks: (i) they form longer sequences of basic instructions, (ii) they shorten repetitive program sequences using `repeat`, (iii) they define their own procedures, (iv) and use these procedures as subroutines, (v) they parameterize their own procedures, (vi) they define and use their own parameterized subroutines.

More information about the currciulum and its specific contents are provided in REF (Hromkovic, 2010).

The following section presents the programming environment XLogoOnline and its approach to error handling dedicated to the above curriculum.

4. Error Handling in XLogoOnline

The XLogoOnline programming environment aims at students' autonomous error recovery by providing assistance in three domains: (i) the environment proactively diagnoses structural errors in text-based Logo programs, (ii) it automatically detects logical errors thanks to a task specification system with integrated solution verification, and (iii) it provides a debugger for students to investigate logical errors on their own.

4.1. *Reporting Structural Errors*

We refer to structural errors as any error that causes the execution pipeline to terminate unexpectedly; be it syntactic errors (which are already apparent during the construction of the parse tree), semantic errors (such as naming errors, missing or redundant arguments. Note that these programs parse legitimately but then fail during interpretation), or more general runtime errors (e.g., type errors, index errors, or other problems that are detected at runtime).

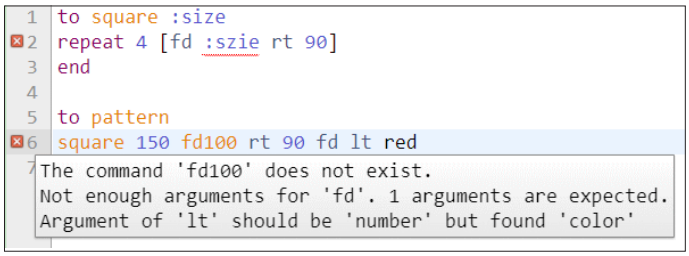


Fig. 3. An example of how XLogoOnline visualizes errors in the environment. All of these cases can be detected statically.

XLogoOnline follows the philosophy of reporting structural errors proactively, that is as early on as possible. For this purpose, the program text is continuously parsed on every keystroke in order to immediately detect syntactical errors in the parse tree. Moreover, the environment tries to turn as many runtime errors as possible into static errors that can be detected before the program is executed. This can be achieved using static program analysis and combined with the check for syntactic errors.

All detected errors are localized in the source code (i.e., the respective token stream) in order to find the exact position in the program text, see Fig. 3. Afterwards, the corresponding text is visually highlighted in the editor and a corresponding error message is added. In the formulation of error messages we make sure that the language is understandable (that is, we use few words that are written in the language learners are acquainted with from the curriculum) but we also take care of formulating error messages consistent throughout all cases.

4.2. *Detecting Logical Errors in Turtle Graphics*

Logical errors, unlike structural errors, do not cause the execution pipeline to fail, but rather produce unexpected results. Such errors can arise in any contexts and must be handled differently than structural programming errors. In fact, what may look like an error in one context may be intentional in another. That is, without insight into the specific objectives of a given program, it is not possible to discern correct from incorrect solutions.

In order to still automatically detect logical errors, XLogoOnline provides predefined tasks with exact specifications of permissible solutions (Staub *et al.*, 2021) (user manual provided in the Appendix). Several grid cells can be connected in a navigation task in which one or more *target cells* are to be visited in any or a predefined order. Additionally, certain cells can be forbidden and also the commands available in a solution can be restricted. Moreover, even the available command set can be restricted by a task. A formal specification allows to discern correct solutions from incorrect ones (see Fig. 4).

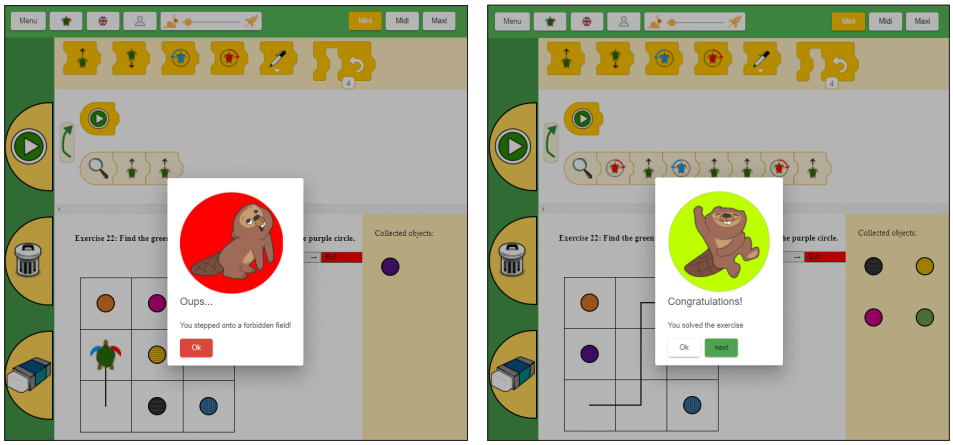


Fig. 4. Two examples illustrating how XLogoOnline automatically detects correct and incorrect solutions in grid-based navigation tasks.

4.3. *Resolving Logical Errors in Logo*

Although it is possible to detect logical errors automatically if the objective and the specification of an admissible solution are known, automatically locating logical errors is neither easy nor didactically desired. The problems used in our curriculum deliberately allow more than one solution. For example, Fig. 5 shows a problem in which a given picture is to be drawn without making right turns. In our experience, this task elicits multiple different solution strategies (e.g., Fig. 6), that all adhere to the given condition and solve the problem. This flexibility should be maintained in order to encourage the exchange of ideas within the class and the comparison of different strategies.

Due to the numerous possible correct solutions and individual ways of thinking, we do not intend to solve the localization of logical errors automatically. By making

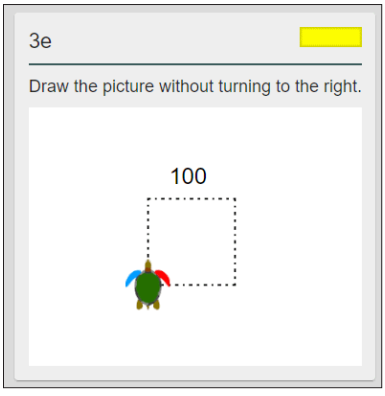


Fig. 5. Sample task.

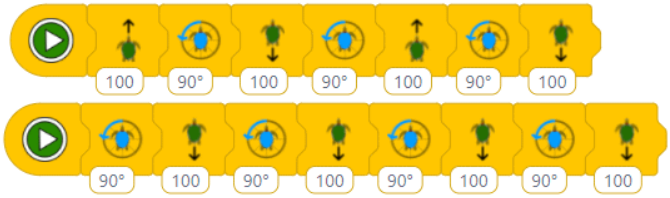


Fig. 6. Two solutions that are equivalently valid.

students solve logical errors independently, they can gain insights that would otherwise be denied to them. In this sense the last tool presented here pursues a different objective than the previous two we present a debugger that enables programmers to fix logical errors on their own.

XLogoOnline provides a reverse debugger, which allows to manually analyze erroneous programs one step after another. At the push of a button, an instruction is executed, allowing the user to compare his or her mental image of program execution with reality. Based on this experience, novices can draw conclusions about the location and the nature of an underlying logical error. Using a simple stack, the program state can be stored in each step allowing previous stages to be reached easily and enabling the course of an error to be replayed as often as desired.

5. Conclusion

For several decades, our community has been using block-based learning environments to provide beginners with a smooth start into programming. There are various reasons why block-based environments are useful: some (like ourselves) see a potential to reach young children who would otherwise struggle with writing. Others, meanwhile, consider structural programming errors a threat for all programmers, independent of their age and experience. Consequentially, opinions also diverge on the question when to switch from block- to text-based programming. Some suggest that blocks should be used well into tertiary education, while we argue that there is no need to stick with block-based environments for so long. We showed an approach of error handling that is employed in the XLogoOnline programming environment and which encourages the autonomous handling of programming errors; both logical and structural ones.

Various text-based learning environments for novices have a *reactive* approach to handle errors. That is, they report errors only during runtime. This decision causes a long and oftentimes frustrating process to start once execution begins: for each fixed structural error, programmers need to re-execute their code, possibly receiving yet another red flag which needs to be fixed before starting all over again. We argue that a proactive approach to error handling can help students skip over this tedious phase more quickly. Our approach allows structural errors to be located and reported at compile time which may lead to a majority of all structural programming errors to be detected and potentially resolved before execution even starts.

References

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-d tool for introductory programming concepts. *Journal of computing sciences in colleges*, 15(5), 107–116.

Dagienė, V., Hromkovic, J., and Lacher, R. (2021). Designing informatics curriculum for k-12 education: From concepts to implementations. *Informatics in Education*, 20(3), 333–360.

Ettinger, A.B. (2012). Programming robots in kindergarten to express identity. *Industrial Engineering*, 2012.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, 18(2), 93–116.

Forster, M., Hauser, U., Serafini, G., and Staub, J. (2018). Autonomous recovery from programming errors made by primary school children. In: Sergei N. Pozdniakov and Valentina Dagienė, editors, *Informatics in Schools. Fundamentals of Computer Science and Software Engineering*, volume 11169, pages 17–29, S.l. Springer. 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018; Conference Location: St. Petersburg, Russia; Conference Date: October 10–12, 2018.

Gugerty, L. and Olson G. (apr 1986). Debugging by skilled and novice programmers. *SIGCHI Bull.*, 17(4), 171–174.

Hromkovic, J. (2010). *Einführung in die Programmierung mit LOGO*, volume 206. Springer.

Hromkovič, J. and Kohn, T. (2018). Einfach informatik 7–9: Programmieren. sekundarstufe i. begleitband. *Einfach Informatik*.

Hromkovič, J., Kohn, T., Komm, D., and Serafini, G. (2016). Examples of algorithmic thinking in programming education. *Olympiads in Informatics*, 10(1–2), 111–124.

Hromkovič, J., Kohn, T., Komm, D., Serafini, G., *et al*. (2017a). Algorithmic thinking from the start. *Bulletin of EATCS*, 1(121).

Hromkovič, J., Serafini, G., and Staub, J. (2017b). Xlogoonline: a single-page, browser-based programming environment for schools aiming at reducing cognitive load on pupils. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 219–231. Springer.

Kohn, T. and Manaris, B. (2020). Tell me what's wrong: a python ide with error messages. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1054–1060.

Komm, D., Hauser, U., Matter, B., Staub, J., and Trachsler, N. (2020). Computational thinking in small packages. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 170–181. Springer.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4), 1–15.

Menta, R., Pedrocchi, S., Staub, J., and Dominic Weibel, D. (2019). Implementing a reverse debugger for logo. In: Sergei N. Pozdniakov and Valentina Dagiene, editors, *Informatics in Schools. New Ideas in School Informatics*, volume 11913, pages 107–119, Cham, 2019. Springer. 12th International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP 2019); Conference Location: Larnaca, Cyprus; Conference Date: November 18–20.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., USA.

Repenning, A. and Ioannidou, A. (2006). Agentcubes: Raising the ceiling of end-user development in education through incremental 3d. In: *Visual Languages and Human-Centric Computing (VL/HCC'06)*. IEEE, p.p. 27–34.

Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M.L., Minsky, M., Papert, A., and Silverman, B. (2020). History of logo. *Proceedings of the ACM on Programming Languages*, 4(HOPL), 1–66.

Solomon, P. (1993). Children's information retrieval behavior: A case analysis of an opac. *J. Am. Soc. Inf. Sci.*, 44, 245–264.

Staub, J. (2021). *Programming in K–6: Understanding Errors and Supporting Autonomous Learning*. PhD thesis, ETH Zurich, Zurich.

Staub, J., Chothia, Z., Schrempp, L., and Wacker, P. (2021). Encouraging task creation among programming teachers in primary schools. In: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 135–146. Springer.

Trachsler, N. (2018). Webtigerjython-a browser-based programming ide for education. Master's thesis, ETH Zurich.

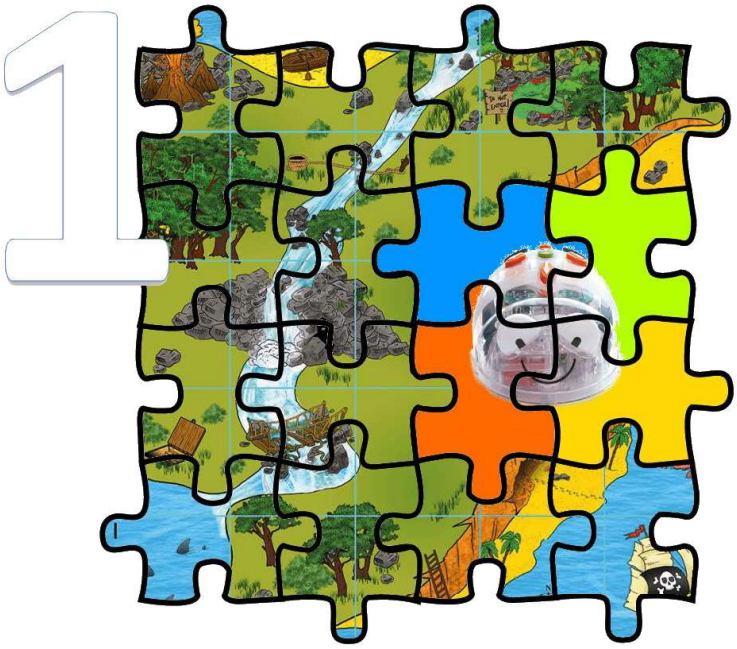
Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM*, 62(8), 22–25.

J. Staub holds the chair of Computer Science and its Didactics at the University of Trier. She studied computer science at ETH Zurich and completed the teaching degree for high school level at the same university. Starting in 2016, she conceived the programming environment XLogoOnline as part of her doctoral studies, which is designed as a tool to explore errors in programming education with novices and to enable the teaching of computer science concepts in a spiral curriculum from kindergarten to high school. XLogoOnline is a learning platform that is currently offered in seven different languages and is now used in schools around the world. Prior to her appointment at the University of Trier, she was a postdoctoral researcher at the Ausbildungs- und Beratungszentrum für Informatikunterricht at ETH Zurich and worked as a lecturer at the University of Teacher Education Graubünden (PHGR). As part of her work at the University of Trier, she promotes teacher education in computer science in the Rhineland-Palatinate area, continues the development of the programming environment XLogoOnline, and uses it to study error handling among programming novices.

Appendix

Exercise Creation in XLogoOnline & LogoOlympia

User Manual

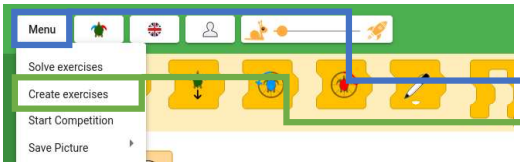


This Document serves as a reference guid for educators using XLogoOnline Mini or the built-in competition mode LogoOlympia

Contents

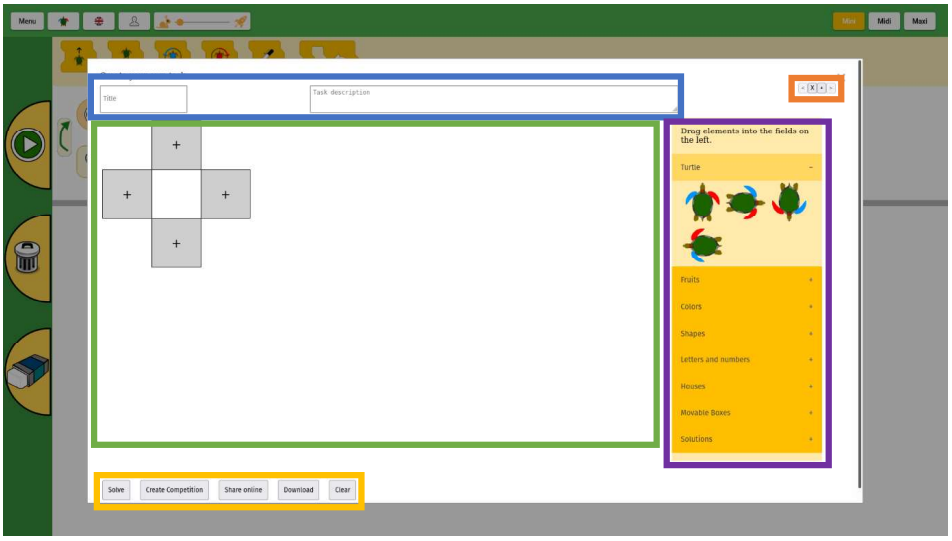
1. Create Exercises.....
2. Validate Exercises .....
   2.1. Solution.....
   2.2. Constraints .....
3. Store and Load Exercises .....
4. Competition.....
   4.1. Score Board and End Competition.....

1. Create Exercises



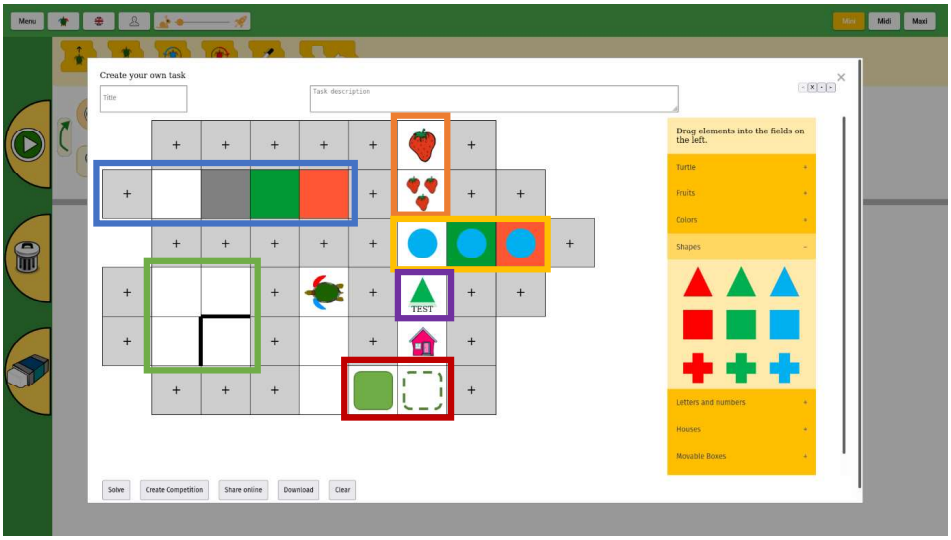
To create a new exercise (or edit a previously loaded one) you first need to open the Menu and then click the "Create exercises" button

This brings up the "Teacher-Tool Dialog".



- The ▭-area holds the exercise title and description.
- The ▭-area represents the grid, on which exercises can be solved.
- The ▭-area allows to add and remove exercises as well as to switch between different exercises.
- The ▭-area presents the options on how to access or share an exercise. A detailed explanation can be found in chapter 3.
- The ▭-area holds the turtle as well as other objects that can be placed on the grid. Furthermore, validations can be configured here, we will explain them in more detail in chapter 2.

2. Validate Exercises



- The tiles in the -area from left to right are:
    1. new tile [+].
    2. default tile [white].
    3. forbidden tile (visible to the student) [grey].
    4. target tile [green].
    5. forbidden tile (not visible to the student) [red].

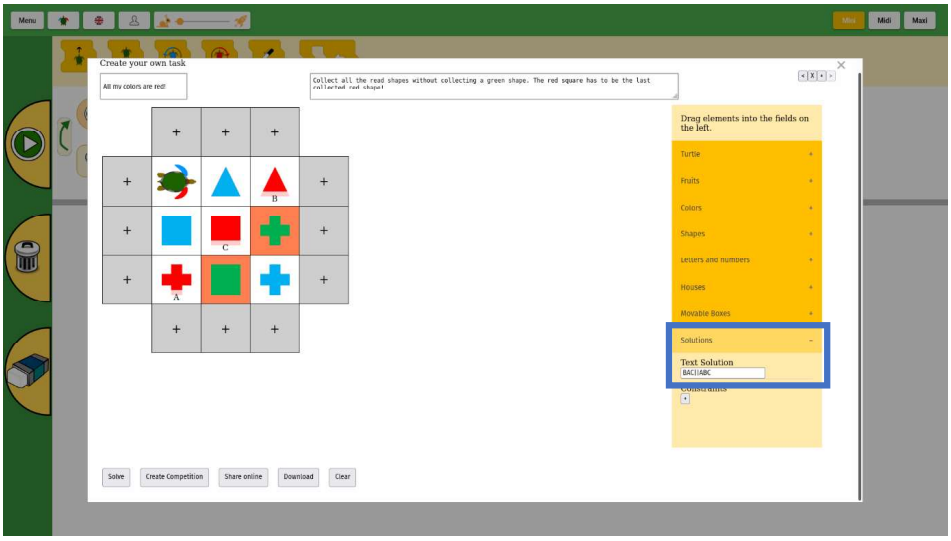
  Stepping on a forbidden tile, results in a failure state, while stepping on a target tile, results in a success state.
  By clicking on a tile, you can cycle through the five states a tile can be in.

- In the -area two “walls” are set up on the top and left of the bottom right tile. If the turtle walks through a wall (in any direction).
- The -area holds 3 times the “blue color” object. From left to right, they were placed on a default tile (2), the second one on a target tile (4) and the last on a forbidden tile (5).
- The object in the -area has a value assigned to it. By default, every object has a value of 1, by clicking on a tile, you can assign a specific value to it. As soon as any object has a value assigned (that is not a number) all other objects will have the default value “” (empty string).
- The -area holds multiple instances of the strawberry object. The strawberry object is special, as it has a default value equal to the number of strawberries depicted on

the object. The strawberries range from one to four strawberries and thus a default value between one and four.

- The ■-area holds a moveable box (the filled-out object on the left) and a target box (the right box with the dashed outline). The moveable box can be pushed by the turtle and the goal is to push a moveable box on every target box. Once this is achieved, a success state is reached and the constraint get checked.

2.1. Solution



In the “Solution” tab, in the “Text Solution” input field, a number can be entered, which then will work as a sum. The values of all the collected objects, will be summed up and compared to the target number. If the sum of all collected objects matches the solution, a success state is reached and the constraints are checked.

If instead of a number, a string is entered into the text solution field, the values of the collected objects will be concatenated and compared to the given solution. If they match a success state is reached and the constraints are checked. In this mode, multiple possible solutions can be separated by “||”. In the given example above, both “BAC” as well as “ABC” would be accepted as solutions. Another option would be to make the solution “AAC” and give both the red cross and the red triangle the value “A”.

2.2. Constraints

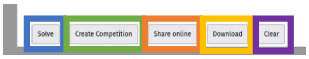
The [X] button can be used to remove a constraint set.

With the [+] button a new constraint set can be added. If a given program passes any constraint set, it will be considered valid by the system.

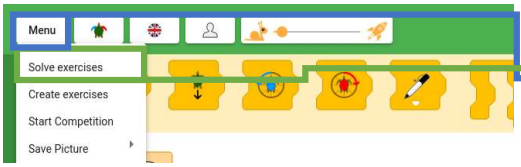
Constraints can be used to limit or ban the use of certain command or all of them. The options for a single constraint are “less than”, “less or equal than”, “equal”, “not equal”, “more or equal than” or “more than”. To ban the use of a command, you can set the constraint type to “equal” and the amount to zero, as in the example has been done to the “Forward” command. To limit the use of commands altogether, a constraint can be put on “Total Commands”. In the example in total up to 4 commands can be used, but none of them can be the forward command.

3. Store and Load Exercises

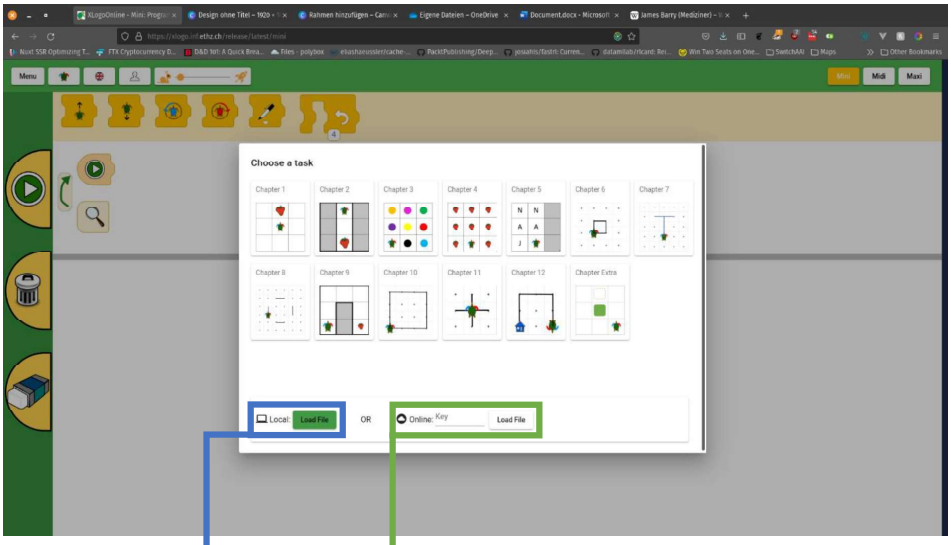
Storing exercises can be done from the teacher tool, with the buttons in the bottom left.



- The -button loads the current exercises in the exercise mode to be solved in the browser. By opening the teacher tool again, the exercises can be modified further and/or saved.
- The -button creates a competition of all exercises and presents you with a competition- and storage-key. The storage key is needed to load the exercises in the future for further modification, while the competition key is needed by participants to enter a competition.
- The -button will save the exercises to our server and present you with a storage key, you can use in the future to retrieve the exercises again.
- The -button can be used to download the current exercises to your computer. They can be uploaded to XLogoOnline on a later date.
- The -button lets you reset the current exercise if you have created multiple exercises the others will not be affected.

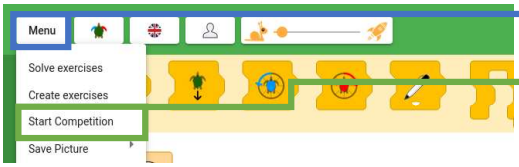


To load an exercise you have to open the “Solve exercises” dialog from the menu



You can either upload one or saved files or load them from the server by specifying one or more storage keys separated by commas.

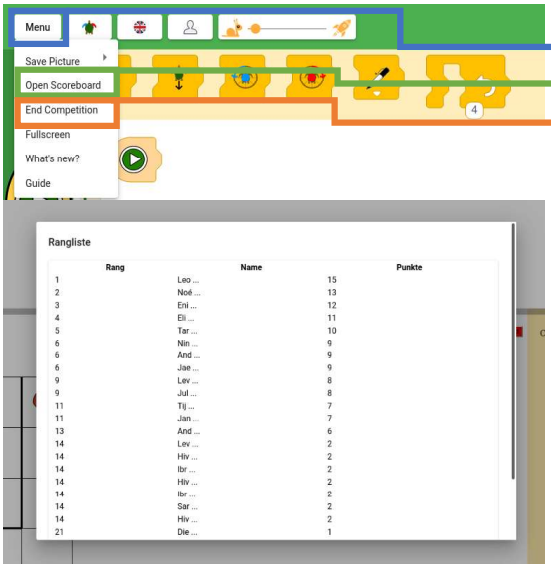
4. Competition



To participate in a competition you need to open the menu and select "Start Competition".

Next you need to enter the competition key.

4.1. Score Board and End Competition



When you are entered in a competition the menu contains an "Open Scoreboard" as well as "End Competition" option.

The "End Competition" button will end the competition for the participant.

The "Open Scoreboard" button will pull up the current competitions score board

Olympiads in Informatics, 2022, Vol. 16, 75–87
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.07

Methods of Tracks for Training Juniors in Olympiad Informatics: The ISIJ Experience

Marina S. TSVETKOVA[1], Vladimir M. KIRYUKHIN[1],
Nikolai A. BORISOV[2], Mikhail I. KINDER[3]

[1]*Russian Academy of Natural History, Russian Federation*
Moscow, 105037, box 47
[2]*Lobachevsky University, Russian Federation*
Nizhny Novgorod, 603022, Gagarina av., 23, 2 bld., 320
[3]*Kazan Federal University, Russian Federation*
Kazan, 420008, 35 Kremlyovskaya str., 2 bld., 806
e-mail: {ms-tsv, vkiryukh, n.a.borisov, detkinm}@mail.ru

Abstract. The article describes methodological approaches to the formation of olympiad competencies for juniors, children 12–15 years old, based on five training tracks in olympiad informatics in accordance with the IOI Syllabus. An integrated approach to the preparation of rounds for children in the framework of the experience of the International School in Informatics for Juniors (ISIJ) is also described.

Keywords: talented children, olympiad informatics, competencies of olympiad informatics, digital skills, computational thinking, algorithmic thinking.

1. Introduction

In the modern world, the computer has absorbed the complex of human knowledge – mathematics, formal logic, combinatorics, physics and circuitry, the theory of algorithms and programming languages, modeling and design tools, and continues to absorb all the new modern scientific knowledge of natural sciences, philology, art ... The computer also gave rise to a new view of man on the world, things and processes, made it possible to create artificial virtual worlds and artificial intelligence for the world of people, things and processes ... Such a broad view of a new kind of human thinking, like computer thinking, is based on algorithmic, formal and logical thinking, research and design thinking, which together created the phenomenon of a new civilization: computer and programs as a new computer style of thinking based on algorithmic thinking (Tsvetkova M., Kiryukhin V., 2021).

Such an artificial digital world is created by humanity on a gigantic scale and pace and is controlled by programs that are also created by people based on artificial intelligence algorithms and significantly change the picture of the world. As a result, life in the digital world has included digital literacy in the educational context, complementing traditional literacy, and requires the formation of computer thinking as a complex of different types of thinking, especially in children.

Children who are passionate about computer science and programming essentially become pioneers of computer thinking at each new stage of its development. But it is necessary to form such a level of a new style of thinking in an integrated manner. At an early level of the formation of computer thinking, it is necessary to promote learning along several training tracks: in mathematical informatics as a formal tool for describing algorithms, in algorithmic thinking as an intellectual tool for the creative creation of algorithms, in programming technique as a computer tool for implementing algorithms, and in the speed of thinking in integration with technological computer skills, that is, to combine the high-speed qualities of human creative work and the high speed of a computer. All this is the basis for the development of integrated approaches in the methodology for the development of talented schoolchildren in olympiad informatics and the development of new types of competitions that will activate various aspects of computer thinking in key competencies included in IOI Syllabus (IOI Syllabus, 2019).

2. Training Tracks of Olympiad Informatics

The experience of holding the International School of Informatics for Juniors, children from 12 to 16 years old (Tsvetkova M., Kiryukhin V., 2018), revealed certain deficiencies in the preparation of children and showed us new directions of training in olympiad informatics, taking into account the international experience of holding Olympiads in Informatics. Juniors generally do not yet have the experience of high olympiad achievements, but they have high motivation in development. It is important to build training tracks for children so that they evenly master the entire range of competencies.

To determine the preparation tracks, consider the IOI Syllabus, which includes the main sections of preparation for olympiad informatics (Kiryukhin V., 2007):

- Mathematical informatics.
- Algorithms.
- Programming technology.
- ICT tools.
- Modeling.

Traditionally, when preparing for Olympiads in Informatics, the emphasis is mainly on algorithms and methods of their implementation in programs, that is, on the development of algorithmic thinking. You can call this approach specialized, but for the preparation of children, it greatly narrows their development horizon. We consider the development of talents more broadly and requires the formation of computer thinking based on algorithms. Juniors do not yet have complex competencies like students. At the

same time, the mathematical foundations of informatics and programming technology are mainly included in the school informatics course in general, which is not enough for the olympiad preparation.

The success of the preparation is expressed in the results of the participation of juniors in the Olympiads in Informatics. This success requires instrumental readiness, which is determined by the level of proficiency in programming tools and technical proficiency, which is determined by the digital literacy of a junior (Tsvetkova M., Kiryukhin V., 2020). But schoolchildren do not have the opportunity to gain experience in competitions in these sections of the olympiad preparation.

In this regard, it is important to supplement the olympiad preparation with new competitive rounds, which will be able to focus on certain important competencies of juniors. Taking this into account, the complex of competencies of olympiad informatics can be represented by five main training tracks. Chief among them is *intellectual* (algorithms, traditional rounds of olympiad informatics). Additional tracks by competency categories are *formal* (mathematical informatics), *instrumental* (programming language), *technical* (computer literacy) and *applied* (modeling various processes and objects by means of algorithms and programs). Additional tracks determine an important aspect of the development of juniors, since this type of training shows them digital samples in various professional fields (technology, science, art, culture, economics, etc.), but using algorithms and programming.

3. Types of Training Rounds for Juniors Based on ISIJ Experience

Mathematical Informatics and programming language are the main creative tools, characteristic of algorithmic thinking. The development of algorithmic thinking is implemented within the framework of training programs on relevant topics from the IOI Syllabus, in particular from the sections Computing Science, Algorithms and Complexity (IOI Syllabus, 2019).

All international Olympiads for schoolchildren and most national Olympiads in Informatics reflect this olympiad track. At ISIJ, this corresponds to a traditional round, called Marathon. The duration of this round is 4–5 hours, and it offers 3 problems for solving (experience of IOI, EJOI, IATI, APIO and other Olympiads).

The basis or foundation of knowledge at the start of preparing juniors for Olympiad in Informatics is formal thinking and instrumental skills. Formal thinking develops from elementary school based on mathematics, and further should be improved in mathematical informatics in the course of school informatics. It is important to form a deep knowledge of the mathematical foundations of informatics, an understanding of optimal solutions, the formation of a fast algorithmic search for ideas through mathematical problems in informatics to enter the olympiad achievements.

Formal thinking based on mathematical informatics can be developed using mathematical rounds as part of the olympiad preparation. Tasks that can be used in the design of Math- round must comply with the Mathematics section in IOI Syllabus (IOI Syllabus, 2019).

Instrumental competence of proficiency in a programming language based on developed formal thinking and an intellectual knowledge stock of algorithms are the keys to the olympiad readiness of juniors. Training in instrumental programming competencies requires separate training for children, taking into account the specifics of the programming environment and the requirements of a particular Olympiad. Therefore, trainings on programming techniques, as basic programming skills, taking into account the specifics of a particular programming language, are very important for children. It is important to choose a language as a basis, where you can show different aspects, approaches for developing code and optimizing it based on the software tool of a particular Olympiad.

It is impossible to involve children in the Olympiads without deep training in programming techniques. The participant of the Olympiad in Informatics should not experience barriers in technical competence in the competition. For this purpose, during the ISIJ, a Coder round is introduced, which reflects this training track. Sections of preparation on which instrumental rounds on programming techniques can be formed are included in the IOI Syllabus, in the sections 6.1 Programming Fundamentals and Software Engineering (IOI Syllabus, 2019).

It is also believed that the participants of the Olympiad must have excellent knowledge of the general competencies of digital literacy (information, computer, communication), that is, masterly master keyboard input, control the interface on a computer, use a computer network and file system ... Sections of training corresponding to the formation of computer literacy are included in IOI Syllabus, in the sections Software Engineering and Computer Literacy (IOI Syllabus, 2019).

For juniors, the requirement to have excellent general competencies in digital literacy can be a barrier to successful performance at the Olympiad. Therefore, trainers need to pay special attention to digital literacy of children, since it removes technical and psychological barriers in the speed of children's work on a computer and in making decisions based on the limitations of a computer or operating system when working with digital information.

All children acquire important digital skills of elementary computer literacy in school. These skills are technically competent high-speed input, fluency in the interface, knowledge of the technical limitations of computer devices and experience in customizing programs for the user, using network services and settings. Skills of working with digital information are the basis for decision-making for data processing, the choice of data structures, taking into account the parameters of the problem, which allows a junior to freely navigate in terms such as memory capacity, computation speed. All this should be included in trainings on mastering the tools of olympiad informatics. In ISIJ, such rounds are called speed work technique rounds based on typical IOI olympiad problems or IOI relay race as *the Estafette* round. The most useful for olympiad preparation here is the C++ programming environment. As tasks for these rounds, it is advisable to use the set of tasks of the IOI archive. A rich collection with an accessible debugging environment, which is successfully used in the ISIJ, is presented on the YandexContest 2022 website (YandexContest, 2022).

The ISIJ experience has allowed us to expand the range of junior competition rounds that make up the set of all-around ISIJ Summer Cup. This helps schoolchildren at an early age evaluate different aspects of training in olympiad informatics across all competency tracks, as well as try to apply their skills in algorithm development and programming in an applied environment. Such an environment can be an environment for simulating the work of a virtual (screen) or real robot. For the ISIJ, this is the *Robot* round.

4. ISIJ Cup Structure

The ISIJ Cup started in 2018 following the EJOI. The order of this Cup is significantly different from the traditional Olympiads in Informatics.

Firstly, it introduces an element of competition in the conduct of the ISIJ, which is very important for the involvement of juniors in the olympiad movement in informatics and for them to get competitive practice.

Secondly, it is held in a hybrid form, that is, both offline and online, which allows junior children who do not have the means for mobility to take part in this competition. As a result, not only European participants take part in the Cup, but also juniors from China, New Zealand, Mongolia, Sri Lanka, the Russian Far East and Siberia. It is also important that when it is carried out in the online form, the large difference in the time zones of residence of the juniors is taken into account.

Thirdly, the variety of rounds within the cup allows to reveal different abilities of all participants and to reward many of them for different manifestations of talent.

Fourthly, all participants in the competition are divided into two groups for summing up: beginners and advanced. This allows children, even with little experience in olympiad preparation, to get the opportunity to participate in an international competition and gain valuable experience of participation in Olympiads in Informatics.

And the last one, the teams of the participating countries work at the school together with their coaches, who also take part in all rounds of the Cup, and their results are evaluated in the same way as for children. This allows coaches to deeply analyze the specifics, complexity and algorithms for solving problems, see the difficulties of their juniors and properly organize training for them to move forward.

By tradition, the ISIJ Cup has the following 5 rounds:

- *Marathon*.
- *Math*.
- *Coder*.
- *Estafette*.
- *Project*.

Let's consider each of these rounds in more detail.

Marathon round is a round for the development of algorithms and programming with individual credit. The round is conducted using 3 problems for groups A (advanced) and B (base) and is designed for 4 hours. Compared to traditional IOI round, the duration of this round is 1 hour less, which gives the participant the opportunity to try to solve

the problems of the round in less time, so that at the IOI there is 1 hour of time for the participant's "internal clock".

Math round is a blitz round on the speed of solving problems in mathematical informatics and logic, which is very important for participants in Olympiads in Informatics. This round offers 12–15 blitz ideas for 2 hours. The round includes 12 blitz problems (for group B) and 15 blitz problems (for group A). All tasks are related to mathematical informatics and computational algorithms. Each problem has one correct solution, and the round is held in the Yandex-Contest system.

Coder round is a round of correcting the program-solution in C++, proposed by the jury for each of the 3 problems. The participant must identify possible problems in each program-solution and correct any mistakes. Problems may reflect upgrades, fixes, additions of part of the algorithm in the proposed programs of the jury. The round is designed for 3 hours and is conducted in the Yandex-Contest system.

Estafette round is a round in which two approaches are implemented: the traditional one for the speed and completeness of solving the problem, the second is the team one. The round involves teams from each country, consisting of no more than 6 juniors. Each team is given as many tasks as there are participants to solve within one hour, but each participant solves only one task. All tasks are IOI tasks or similar. At the end of the round, each team is awarded an average score of the team (the total score of the team divided by the number of participants of this team). This round can also be conducted on problems that are new to the participants, but it is better to use a deep study approach, when the participants have already solved the problems, analyzed the solution and the *Estafette* round is needed to check how the children cope with the known problem for the speed and completeness of the solution. It is important that the child's internal clock and memory are included during the round.

Project round (*Robot* round) is devoted to the topic of programming in C++ for unmanned devices for various purposes. The round is conducted for teams of 2–3 participants online using virtual or real robots or in person using robotic equipment. The *Robot* round is focused on the STEM approach, and its goal is to show how you can apply your algorithms and programming skills to applied information technology. A *Robot* round can deal with various applied topics, for example, cybernetics and artificial intelligence, and in the process of carrying out it can be used devices with feedback and sensors, learning devices, groups of interconnected devices (a swarm of drones), moving models in different environmental conditions, models manipulators and others.

Such a structure of the ISIJ Cup implements all five tracks of the olympiad preparation of children, and can become traditional in the work of coaches, both in the formation of deep complex olympiad competencies of children, necessary to participate in Olympiads in Informatics, and in the improvement of the technique of speed thinking and complex digital skills, which help in the development of computer thinking in a broad sense.

Next, we will consider in more detail several features of some of the elements of the ISIJ Cup structure described above.

5. Features of the Math Round

The purpose of the math blitz round is to motivate the participants of the ISIJ to study sections of mathematics as part of the preparation for olympiad informatics, as well as check the preparation of participants for the main topics of mathematical informatics.

Each math blitz round task is a small mathematical olympiad problem on one of the topics of the school curriculum of the course "Mathematical Foundations of Informatics": logic, combinatorics, set theory, graphs, elements of probability theory, chess, numerical laws and sequences, number systems, computational, geometric algorithms and strategies, etc.

The set of tasks includes such a number of tasks on various topics, which, according to the developers, should take an average of 2 hours for their complete solution. According to the complexity of the problem, the jury takes into account the approximate time for the blitz. Blitz solution orients the student towards a creative approach and solving the problem "in the mind".

The examples of tasks on specific topics below show how a task is presented to a participant and how he should submit a response for review.

The participant receives the text of the problem and a description of the format of the response. An automatic check system records his correct answer. After the end of the round, the participants of the competition are given access to solutions of problems. Below are examples of some of the math blitz round tasks.

Task "Cake"

The cake has the form of a parallelogram with vertex coordinates (0; 0), (4; 0), (6; 6), (2; 6). Rabbit and Fox share the cake as follows. Rabbit points to a point on the surface of the cake, and the Fox cuts the cake into two pieces in a straight line passing through this point and takes one for himself. Everyone wants a bigger piece. Where should the Rabbit put the dot?

Output the answer – *the coordinate of this point as two numbers separated by spaces.*

Answer: 3 3.

Solution. This is the center of the parallelogram, i.e., intersection point of diagonals. Rabbit can't get more than half of the cake. Any straight line passing through the center of a parallelogram divides it into two equal parts.

Note. This is a simple geometric problem on the properties of a parallelogram and on the topic "Game strategies". The approximate time for solving the problem is 3–5 minutes.

Task "Sum of permutations"

Find the sum of all five-digit numbers that are obtained by permuting the numbers from 12345.

Print one integer that is the answer to the problem.

Answer: 3 999 960.

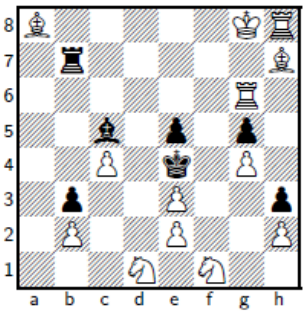
Solution. There are only 5! = 120 ways to rearrange numbers in the number 12345. Among these methods, in exactly one fifth among these (i.e., in 24 cases), the number 1 is in the first place. The same is true for any number and for any place. Therefore, the required sum is equal to:

$$24(10\,000 + 1\,000 + 100 + 10 + 1 + 20\,000 + 2\,000 + 200 + 20 + 2 + 30\,000 + 3\,000 + 300 + 30 + 3 + 40\,000 + 4\,000 + 400 + 40 + 4 + 50\,000 + 5\,000 + 500 + 50 + 5) = 24(11\,111 + 22\,222 + 33\,333 + 44\,444 + 55\,555) = 24 \cdot 11\,111 \cdot (1 + 2 + 3 + 4 + 5) = 3\,999\,960.$$

Note. This is a combinatorial problem on the topic "Permutations", "Number systems". The approximate time for solving the problem is 10–15 minutes.

Task "Chess"

In position on the chessboard, White has only one move that does not checkmate for Black.



The coordinate of a cell on a chessboard is given by a capital Latin letter and a number without a space between them. In the response line *enter the required move in the format of cell coordinates: cell coordinate "from" and cell coordinate "to" separated by a space between them.*

Answer: G6 C6.

Solution. After the move Rg6–c6! the rook blocks the diagonal of the white bishop on a8, and now black has protection from checkmate Rb7:h7.

Note. The approximate time for solving a chess problem is 8–12 minutes.

6. Features of the Estafette Round

This round aims to identify the basic qualifications of school participants in preparation for the IOI. The peculiarity of this round is that the basic qualifications of the participants are determined in the process of teamwork when solving olympiad problems that were offered from the IOI archive of previous years. The basic qualification includes the skill of technically competent and high-speed work with the specifics of IOI tasks, the speed of solving problems on the round (the target is one hour per one task), the strategy for solving the problem on the round (the ability to navigate the parameters of subtasks and skillfully regulate the time for sending solutions to simple restrictions, gaining points). Collections of IOI tasks available for constructing the Estafette round are presented on the Yandex Contest website in the IOI archive in the public domain (YandexContest, 2022).

Teams for participation in the round are formed from participants from one country. No more than 6 juniors can be on one team. For each team, one set of tasks of past IOIs, common for all teams, is randomly allocated, and the tasks included in it can be

of varying complexity. Team members, together with the coach, distribute tasks among themselves so that each team member gets one task, which he can solve in the best way within 1 hour, which corresponds to the duration of the round.

The solution to each task is evaluated similarly to the corresponding IOI task, but the results are summed up on a team-by-team basis. If a participant has completed the task for a full point in less than 1 hour, then his score is increased taking into account the time to solve (the speed coefficient is taken into account). The team score is calculated as the average score of all team members for their tasks. Team round medals are ranked by the teams' average scores at 25, 25 and 50 percent for gold, silver and bronze medals.

Experience has shown that it was the Estafette round that caused the greatest difficulties for ISIJ participants in solving tasks of all rounds of the Cup. Out of 200 participants in 2021, only 2 juniors managed 75–100 points in the relay. Thus, juniors do not have a trained sense of timing, a strategy for solving tasks, as well as high-speed work techniques. Also, the Estafette round showed that the participants are practically not focused on the IOI tasks, do not own the IOI archive, which is an important criterion for the formation of the basic olympiad qualification for participation in the IOI. All this must be taken into account in the further preparation of juniors to participate in the IOI.

7. Features of the Coder Round

The Coder round is designed to identify deficiencies in programming techniques among juniors. To participate in the IOI, it is important to motivate participants to develop their programming skills in C++, to show participants the features of the C++ language, to check their level of proficiency in C++ capabilities for implementing olympiad algorithms, to develop critical thinking for analyzing algorithms in C++, to be able to transform code taking into account the advantages of C++.

The round consists of three tasks (based on three types) for 4 hours. For group A (advanced group), two tasks of type 1 and one task of type 2 are offered. For level B (beginners level), three tasks are proposed, one of each type. The tasks are solved in the Yandex contest system. For a problem of type 1, the solution is the participant's code after correcting the jury's error in the source text (no more than three operators to correct). For a problem of type 2, the solution is the modified by the participant code, which will allow for a given incomplete jury solution to get a solution that can get a full score when tested on all jury tests for this task. As a solution to a problem of type 3, a test is provided that reveals an error in the code for solving the problem assigned by the jury. The test either catches an error or it doesn't.

A problem of type 1 is evaluated as 100 points if the participant's solution is correct and 0 points if it is incorrect. The problem of type 2 is evaluated out of 100 points on the tests of the jury to check the solution provided by the participant. In a problem of type 3, the jury code contains two algorithmic errors. If the participant's test finds one error in the jury's code, then he gets 50 points, if both errors, then 100 points. The maximum score for a Coder round is 300 points.

8. Features of the Project (Robot) Round

The developer of the Project (Robot) round sets the methodology for its implementation in agreement with the International Scientific and Technical Committee[1]. He has the right to determine the type of robot for the round, develop a set of tasks and provide robots for the round venues for the participating teams on the ISIJ Cup site. It is also possible to conduct a round in an online format, while the tasks of the round are focused on managing a virtual (on-screen) command executor. It can be a controlled device, a training robot, a software environment with robot control, a software environment with decision-making to control the device. The online round is open to all registered ISIJ participants.

Teams of 2–3 juniors participate in the round, the teams are divided into groups A (advanced) and B (basic). The round offers 2–3 tasks. Each task may contain simpler subtasks or a set of steps for executing control commands. The duration of the round is 4 hours, and it is carried out on robots that are the same for all teams. The solutions of the problems of the round are programs in the C++ programming language. During the round, each participating team gets access to the robot and computer and downloads the solutions created by the team into it, either in the presence of a coach at training rounds, or in the presence of a jury, or under video recording on a real round. The round is held in the competition hall with working points for the participating teams indicating the number, group A or B and the country.

Tasks can include valid task types with subtasks in groups A and B.

Task types can be:

- The task of simple control of the movement of robot elements on the test site, indicating all possible obstacles or conditions for performing actions by the robot, taking into account the parameters of the objects with which the robot performs an action.
- The task of selecting/search for types of movement on a given track from target 1 to target 2 with possible conditions, a set of valid commands and feedback with checking the valid actions of the robot.
- The task of controlling a robot under conditions of uncertainty, taking into account the restrictions imposed in the task.
- A task of increased complexity (for group A) to adjust the control of the robot with feedback – a response to an action in real time with a given clock delay.

The solutions of each task of the Robot round are evaluated out of 100 points, but complex tasks for additional subtasks for group A can be evaluated at 200 points. The maximum score for solving the tasks of the round is 300 points. To obtain a rating distribution, it is necessary to differentiate solutions by steps or subtasks. The awarding of teams following the results of the round fully complies with the awarding rules defined in the Regulations on Online rounds of the ISIJ Cup.

[1] www.isi-junior.com

9. Conclusion

Monitoring of the performance indicators of the ISIJ 2018–2022 participants in the Olympiads in Informatics showed that optimally two years of immersion of children in the training tracks described above is enough to create conditions for a jump to the level of high olympiad results. However, this requires strengthening in children a stable motivation for independent work in the tracks of the olympiad informatics, the olympiad culture of training. ISIJ participants from different countries become leaders in national Olympiads in Informatics, and also win medals at various international Olympiads. This is the main value of such an integrated approach in the preparation of juniors, and what is very important, it allows you to unlock the potential of the child through participation in ISI.

However, all children who have completed such training tracks, even without remaining in the Olympiad movement, demonstrate stable skills of computational thinking in the future. This helps them in their individual choice of profession to realize their potential and expands their horizons for the application of programming in different professions, which is extremely valuable.

A unique feature of the ISIJ is the participation of all coaches of the ISIJ teams along with juniors in all tracks. This allows them to pedagogically evaluate the complexity, specificity, content and methodology of each track and then apply the gained experience in their future work. Informatics teachers, coaches of teams from different schools of the world actually go through a seasonal international internship at the ISIJ, get acquainted with the specifics of International Olympiads in Informatics, exchange their work experience with each other and can bring these innovations to work with juniors methodically competently (ISIJ, 2022).

Reference

IOI (2022). *International Olympiad in Informatics*. https://ioinformatics.org/

IOI Syllabus (2019). *IOI Syllabus*. https://ioinformatics.org/files/ioi-syllabus-2019.pdf

ISIJ (2022). *International School in Informatics for juniors*. http://isi-junior.com/

Tsvetkova, M., Kiryukhin, V. (2021). Algorithmic Thinking and New Digital Literacy. *Olympiads in Informatics*, 2021, 15, 105–118. https://ioinformatics.org/journal/v15_2021_105_118.pdf

Tsvetkova, M., Kiryukhin, V. (2020). Top 10 Key Skills in Olympiad in Informatics. *Olympiads in Informatics*, 2020, 14, 151–167. https://ioinformatics.org/journal/v14_2020_151_167.pdf

Tsvetkova, M., Kiryukhin, V. (2018). International School in Informatics "Junior" for IOI Training. *Olympiads in Informatics*, 2018, 12, 187–193. https://ioinformatics.org/journal/v12_2018_187_193.pdf

Kiryukhin, V. (2007). The Modern Contents of the Russian National Olympiads in Informatics. *Olympiads in Informatics*, 2007. 1, 90–104. https://ioinformatics.org/journal/INFOL017.pdf

YandexContest (2022). *Yandex Contest website, the IOI archive*. https://contest.yandex.ru/ioi/

M.S. Tsvetkova, professor of the Russian Academy of Natural Sciences, PhD in pedagogic science, prize-winner of competition "The Teacher of Year of Moscow" (1998). From 2002 to 2018 she is a member of the Central methodical commission of the Russian Olympiad in informatics and the pedagogic coach of the Russian team on the IOI. She is the author of many papers and books in Russia on the informatization of education and methods of development of talented students. She is the author of official textbooks and copybooks in Russia for primary school in Informatics. She is author and director of the International school in Informatic ISIJ (since 2017). She is the Russian team leader (2013–2017). She was awarded the President of Russia Gratitude for the success organizing the training of IOI medalists (2016). She is now the Expert of Committee on Education and Science State Duma of the Russian Federation (since 2017), and she has the Committee on Education and Science State Duma Gratitude (2021).

V.M. Kiryukhin is professor of the Russian Academy of Natural Sciences, PhD. He is the author of many papers and books in Russia on development of Olympiad movements in informatics and preparations for the Olympiads in informatics. He is the exclusive representative who took part at all IOI from 1989 to 2017 as a member of the IOI International Committee (1989–1992, 1999–2002, 2013–2017) and as the Russian team leader (1989, 1993–1998, 2003–2012). He received the IOI Distinguished Service Award at IOI 2003, the IOI Distinguished Service Award at IOI 2008 as one of the founders of the IOI making his long term distinguished service to the IOI from 1989 to 2008 and the medal "20 Years since the First International Olympiad in Informatics" at the IOI 2009. He was the chairman of the IOI 2016 in Russia and has the award medal of the President of Russia (2016) for organizing the Olympiad in Informatics in Russia and training IOI medalists since 1989. He is now the President of the International Organizing Committee of the ISIJ.

N.A. Borisov is associate professor at the Nizhny Novgorod State University (Lobachevsky University), PhD. He is now the Chairman of the Scientific-Technical Committee of ISIJ / member of ISIJ-Cup Jury. He was the leader of Russian Team in EJOI-2017 in Sofia. He is also an active participant of ICPC movement, and the leader of Nizhny Novgorod University student team that won the last ICPC World Championship in Moscow in October 2021.

M.I. Kinder is associate professor of the Institute of Mathematics and Mechanics (Lobachevsky Institute) of Kazan Federal State University, Ph.D. in physics and mathematics. More than 30 years of experience in Olympiad mathematics and computer science, member of the jury of the International ICL-Tournament (Kazan) in programming among students and schoolchildren. Among his students are winners and prize-winners of the All-Russian Olympiads in mathematics and informatics. He is the author of many articles and books in Russia on Olympiad mathematics and computer science. He is also an active member of the ICPC movement. Deputy Chairman of the ISIJ Scientific and Technical Committee / member ISIJ-Cup Jury.

Olympiads in Informatics, 2022, Vol. 16, 89–106
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.08

Detecting Plagiarism as Out-of-distribution Samples for Large-scale Programming Contests

Runfan WU, Aohui LV, Qiyang ZHAO
SKLSDE and SCSE, Beihang University
e-mail: {alralr, luaohui, zhaoqy}@buaa.edu.cn

Abstract. In competitive programming, standard solutions for easy tasks are usually simple and shorter, making submissions more convergent both in idea and texts. The huge difference in submission diversity between easy and hard tasks, brings inescapable challenges to plagiarism judging by means of similarity thresholding. In this paper, by drawing the strong data support from the China National Olympiads in Informatics (NOI), we study the statistical characteristics of submission similarities for tasks of wide range of difficulty degrees. Finally, we propose a new adaptive method to detect submission plagiarism as out-of-distribution samples, together with a large-scale challenge dataset of competitive programming submission plagiarism detection. Our method is shown to be of higher accuracy and robustness, thus feasible and reliable for large-scale competitive programming contests.

Keywords: competitive programming, plagiarism detection, out-of-distribution.

1. Introduction

Programming contests are competitive programming design events, where contestants need to finish source codes fulfilling various resource consumption restrictions, and are expected to make submissions correct with their best effort (Halim *et al.*, 2013). Because of the advantages of objectivity, straightforwardness and relative unbiasedness, programming contests are widely adopted for qualifications and assessments related to computer algorithms.

Cheating is a troublesome issue in competitive contests, including directly plagiarizing the source codes or copying ideas from other contestants. Usually, cheaters are lack of thorough understanding of the plagiarized codes or ideas, making their submissions full of segments which are almost identical to original ones. Therefore, plagiarism detection is a reliable approach to revealing source code cheating. Automatic tools, mainly specifically-designed softwares, are usually adopted in plagiarism detection.

The present situation regarding plagiarism calls for a more accurate, robust, and adaptive plagiarism detection system. Many teachers against source code plagiarism find themselves overwhelmed by the recent surge of the admission counts of computer-related majors, which they barely manage with overreliance on automated plagiarism detection tools (Roberts *et al.*, 2018). For easier tasks with short and less diverse standard solutions, contestants are more inclined to finish similar submissions of identical algorithms and data structures. Most submissions would be highly similar to each other for these tasks, whereas the situations for hard tasks are totally different. This brings a great challenge to traditional plagiarism detection methods based on similarity thresholding (Freire *et al.*, 2007). Furthermore, the open-sourcing of common plagiarism detection tools[1] enables cheaters to crack and evade plagiarism detection.

Cheating codes are usually like stitched monsters -swallowed ideas, code segments migrated from others, exhibiting exceptionally high similarity with original submissions which are plagiarized. Therefore, source code plagiarism detection can be regarded as recognizing out-of-distribution samples, which aligns with the objective of outlier detection (Ruff *et al.*, 2021). We propose to answer current setbacks in plagiarism detection by taking similarity distributions of submissions into account. The main contributions of our research are:

- We propose a robust, accurate, and adaptive source code plagiarism detection method with sufficient accuracy.
- We build a highly automatic plagiarism detection platform for porting related algorithms, supporting batch manual inspection of suspicious code pairs under a predetermined plagiarism filtering ratio.
- We employ a plagiarism detection dataset based on real-world, large-scale data from the Certified Software Professional (CSP) programming contest[2] and including a variety of problem designs and contestant code styles.
- We verify the performance of our method with the OI dataset and discover that our method outperforms conventional plagiarism detection methods.

The structure of this paper is as follows. Section 1 provides an introduction. Section 2 describes the background of this paper and the related works. Section 3 discusses the details and implementation of our method. Section 4 presents the experimental findings. Section 5 concludes this paper and offers outlooks.

2. Backgrounds

In a programming contest, tasks are usually designed to be of various difficulty degrees and distinct skill coverages, thus to examine contestants comprehensively. On the other hand, contestants might be much different in problem-solving ways and capabilities,

[1] Refer to Table 2.1 for details.

[2] The CSP programming contest, part of the qualification process for the NOI, is regarded as part of the Olympiad in Informatics (OI)in China. An overview of the CSP submission dataset is in Table 4.1.

and have distinct coding styles. It makes the distributions of similarities between pairs of submissions varying dramatically across tasks and contestant groups. It is extremely hard to designate a global threshold for all situations in conventional plagiarism detection methods. Implementing an adaptive plagiarism detection method could potentially alleviate the issue of varying distribution, thus significantly increase the reliability and efficiency of plagiarism detection.

With the trends of open-sourcing, most existing plagiarism detection tools have either released the original source, or been re-implemented by third parties. Table 2.1 summarizes the situation. Since those plagiarism detection tools are easy to access, cheaters are able to develop cheating skills in a trial-and-error mode to evade plagiarism detection. On the contra try, when detecting plagiarism as out-of-distribution samples, it is dependent on all submissions which are untouchable for cheaters during contests, thus cheaters cannot crack the detection scheme easily as before.

There are mainly two streams of plagiarism detection methods: intrinsic detection and extrinsic detection (Foltýnek *et al.*, 2019).

2.1. *Intrinsic Plagiarism Detection*

Intrinsic detection links source codes with their authorships, capturing the lack of stylistic distinction stemming from plagiarism. There are two approaches to intrinsic detection with an identical final step (Bandara and Wijayarathna, 2011). One is to straightforwardly decide the author of every source code by the maximum likelihood principle. The other is to partition the approximated distribution of source code features. The common final step is to search for the unmatchedness of the predicted and claimed authorships.

Intrinsic detection has a solid foundation on probability theory and decent interpretability. However, it is not practical in programming contests, where an extreme number of contestants each submit few source codes. The first is prone to the confusion of authors, while the second requires an impractical granularity of partition.

Table 2.1
Source code availability of common plagiarism detection systems

System	Implementation	Source code URL
MOSS (Schleimer *et al.*, 2003)	Third-Party	`https://github.com/agranya99/MOSS-winnowing-seqMatcher`
SIM (Gitchell and Tran, 1999)	Official	`https://dickgrune.com/Programs/similarity tester/`
YAP (Wise, 1996)	Third-Party	`https://github.com/zymk9/YAPDS`
JPlag (Prechelt *et al.*, 2000)	Official	`https://github.com/jplag/jplag`

2.2. *Extrinsic Plagiarism Detection*

Extrinsic detection mainly focuses the relation between one source code or source code pair to the others, instead of the relation between source codes and their authorships. It is more frequently used in programming contests. Different types of extrinsic detection can be characterized by the feature extraction method.

Text comparison-based methods. The main aim is to detect repeating character sequences or any of the derived features, which is complexified text comparison. A relatively representative method is local finger printing algorithms (LFA), which are employed by MOSS and JPlag (Schleimer *et al.*, 2003; Prechelt *et al.*, 2000). Typically an LFA extracts the positionally independent features of every window in the original strings, which partly provides robustness against code fragment repositioning. These methods make a hasty assumption that all edits do not change local features radically, but it is not always true across all scenarios.

Classification-Based methods. Given a source code pair, the state of plagiarism could be codified as two classes. Adding intermediate classes tends to ease the classification of borderline samples. Viewing source codes as character sequences, Arwin and Tahaghoghi (2006) propose using general classifiers for the problem after extracting source code features with a recurrent neural network (RNN). These methods oversimplify group wise relations into pairwise labels, thus are unable to deal with group plagiarisms without modifications.

Outlier detection-based methods. Outlier detection is the process of determining samples distant from its distribution. Such methods assume in-distribution source code pairs as innocent and out-of-distribution ones as suspicious and necessary for further investigation. There are five general steps of source code plagiarism detection based on outlier detection (Foltỳnek *et al.*, 2019). Fig. 2.1 illustrates the definition of out-of-distribution samples[3]. This type of methods are outlined as follows.

2.2.1. *Outline of Outlier Detection-Based Methods*

Data preprocessing. Preprocessing is likely needed before the main steps to remove the portions of source codes that is relatively irrelevant to plagiarism detection. Kamalim and Chivers (2020) acknowledge the need of tokenization for reducing factors not determinative of semantics. Wise (1996) proposes lowercasing all tokens relatively early. Đurić and Gašević (2013) regards high frequency tokens removable, for they barely contribute to source code distinguishability despite providing syntactic conformance.

Feature extraction. There are two major basic ideas for feature extraction. One is to calculate the distance of the feature vectors for every source code. Yasaswi *et al.* (2017) uses RNN, viewing source codes as character sequences. Freire *et al.* (2007) consider

[3] The blue and red points indicates respectively the ordinary samples and outliers, and the circles denote group boundaries.

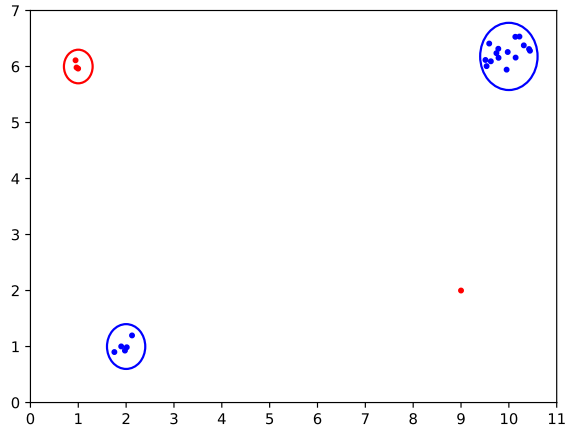


Fig. 2.1. Illustration of out-of-distribution samples and sample groups.

source codes as token streams, extracting the frequencies of each token type as the features. The intermediate representation (IR) (Rabbani and Karnalim, 2017) or the generated machine code (Arwin and Tahaghoghi, 2006) of the source codes could also be treated as regular ones.

The other is to calculate a similarity metric for each source code pair after extracting pairwise features. Freire *et al.* suggest to measure the growth rate of the informational entropy when the two source codes in question are concatenated (Freire *et al.*, 2007). In the perspective of string editing, local repetitive substrings (Karp and Rabin, 1987) and local positional independent features (Prechelt *et al.*, 2000; Schleimer *et al.*, 2003) can also derive the pairwise similarity metric.

Distribution approximation. A similarity matrix could be constructed by the similarity values of each source code pair. Considering each column of the similarity matrix as feature vectors, we can detect the outliers by the approximation of the feature distribution, using support vector machines (SVM) (Suthaharan, 2016) or sparse auto encoders (Ng *et al.*, 2011).

The similarity matrix can also be regarded as an adjacency matrix of a graph, on which an implicit distribution approximation might be performed using graph algorithms to find abnormal nodes or edges. For example, graph embedding is able to transform graph nodes into their vector representations. Frequently used graph embedding algorithms include multidimensional scaling (MDS) (Cox and Cox, 2008), Node2Vec (Grover and Leskovec, 2016), structural deep network embedding (SDNE) (Wang *et al.*, 2016), etc. Fig. 2.2 depicts feature extraction and distribution approximation in conjunction.

Suspicious code pair filtering. For the convenient and easily interpretable quantification of the possibility of plagiarism, many existing methods employ a single suspiciousness index for each source code pair (Devore-McDonald and Berger, 2020; Freire *et al.*, 2007; Ajmal *et al.*, 2013; Yasaswi *et al.*, 2017; Sulistiani and Karnalim, 2019;

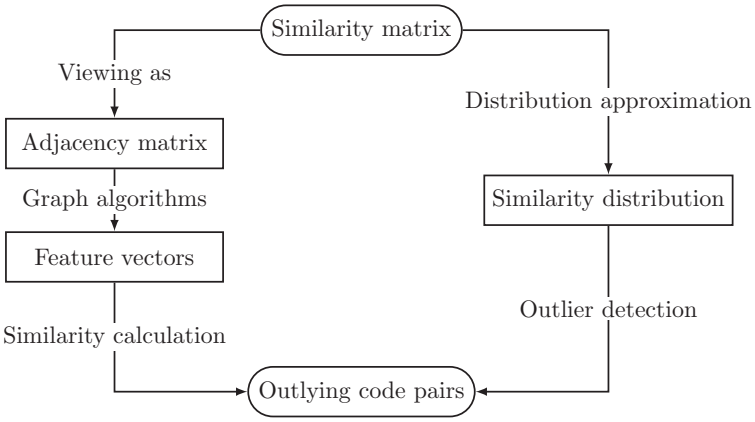


Fig. 2.2. Feature extraction and distribution approximation.

Jiffriya *et al.*, 2014). With the aid of the suspicious index, human operators are able to filter the most relevant pairs for manual inspection (Devore-McDonald and Berger, 2020; Freire *et al.*, 2007). The suspiciousness indices are typically computed from the feature vectors of the individual source codes using vector similarity functions, such as the Euclidean distance (Ajmal *et al.*, 2013; Yasaswi *et al.*, 2017) and cosine similarity (Rahutomo *et al.*, 2012; Sulistiani and Karnalim, 2019; Jiffriya *et al.*, 2014).

Checking and evaluation. Manual inspection results are generally regarded as the reference for determining the status of plagiarism. The commonly used method is to inspect the source code pairs whose ranks of the suspiciousness index are within a previously chosen ratio, then calculate the precision, recall, and F1 score using both the predictions and the manual inspection results (Yasaswi *et al.*, 2017; Flores *et al.*, 2014; Lee *et al.*, 2012).

3. Our Method

3.1. *Data Cleaning*

Data cleaning is the removal of the factors with little relevancy to plagiarism from the submissions. In our method, there are four data cleaning steps executed in succession.

Deletion of irregular submissions. Irregular submissions are those with excessive line count or line width, which typically contains a nonsensical paragraph repeated verbatim innumerably. Fig. 3.1 exhibits an example. The efficiency of plagiarism detection system will be critically impared unless those submissions are removed.

Deletion of submissions with insufficient line counts. Typically those submissions is either a framework or a program that could handle only the cases dispensed with

```
    1  #include <stdio.h>
    2
    3  int main(int argc, char **argv) {
  ...       /* parts related to problem solving */
   49
   50        printf("hello␣world\n");
   51        printf("hello␣world\n");
  ...       /* repetitive content */
10000       printf("hello␣world\n");
10001
10002       return 0;
10003  }
```

Fig. 3.1. An example of irregular submissions.

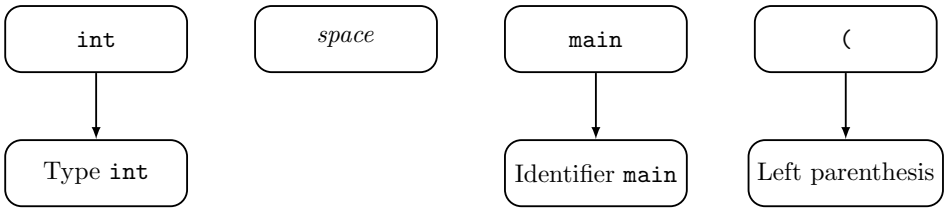


Fig. 3.2. Illustration of tokenization.

the original program description, indicating that the contestant was unable to discover a valid idea. The submissions of both types are extremely inadequate, rendering plagiarism detection on them unnecessary. We also include empty submissions in this step.

Tokenization. Tokenization is to transform the source code from its string form to the token stream form according to the programming language syntax, removing the influences of the factors less relevant to plagiarism. The token stream is the input of the next step, with additional properties preserved, such as the type and function names. Fig. 3.2 illustrates the current tokenization process.

3.2. *Feature Extraction*

We use greedy string tiling (GST) algorithm, which is a widely used algorithm for string similarity (Prechelt *et al.*, 2000). The steps of the GST algorithm are in Fig. 3.3. An optimization of this algorithm is to precalculate the rolling hashes (Karp and Rabin, 1987) of every window of length M on both strings.

Algorithm 1: The GST algorithm

Input: Strings $a = a_1a_2 \cdots a_m, b = b_1b_2 \cdots b_n$, minimal length of valid common substrings M

Output: Similarity s

$tiles, a', b' \leftarrow \{\}, \underbrace{00\cdots0}_{\times m}, \underbrace{00\cdots0}_{\times n};$

do
- $maxmatch, matches \leftarrow M, \{\};$
- **for** $1 \le i \le m$ **do**
  - **if** $a'_i = 0$ **then**
    - **continue**;
  - **end**
  - **for** $1 \le j \le n$ **do**
    - **if** $a'_j = 0$ **then**
      - **continue**;
    - **end**
    - $k \leftarrow 0;$
    - **while** $a_{i+k} = b_{j+k}$ **and** $a'_{i+k} = b'_{j+k} = 0$ **do**
      - $k \leftarrow k + 1;$
    - **end**
    - **if** $k = maxmatch$ **then**
      - $matches \leftarrow matches \cup \{(i, j, k)\};$
    - **end**
    - **else if** $k > maxmatch$ **then**
      - $maxmatch, matches \leftarrow k, matches \cup \{(i, j, k)\};$
    - **end**
  - **end**
- **end**
- **for** $(i, j, k) \in matches$ **do**
  - **for** $0 \le k \le maxmatch - 1$ **do**
    - $a'_{i+k}, b'_{j+k} \leftarrow 1, 1;$
  - **end**
- **end**
- $tiles \leftarrow tiles \cup matches;$

while $maxmatch > M;$

$s \leftarrow 0;$

for $tile \in tiles$ **do**
- $s \leftarrow s + |tile|;$

end

$s \leftarrow \frac{2s}{m+n};$

return $s;$

Fig. 3.3. Algorithmic steps for the GST algorithm.

3.3. *Approximation of Distribution*

We choose graph embedding as the approach for this step, extracting a feature vector for every valid source code. There are two graph embedding algorithms to be used.

Multidimensional scaling (MDS). MDS (Cox and Cox, 2008) is a dimensionality reduction method. It calculates the dimensionally reduced feature matrix $\mathbf{X}'$ from the symmetric pairwise distance matrix $\mathbf{D}$, which is derived from the unknown feature matrix $\mathbf{X}$. If we regard $\mathbf{D}$ as a weighted adjacency matrix, $\mathbf{X}'$ can be viewed as the embedding of the corresponding graph, effectively converting it to a graph embedding algorithm.

Suppose the dimensionalities of the original and dimensionally reduced feature vectors are m and k ($1 \leq k < m$) respectively. We denote $\mathbf{X}$ and $\mathbf{X}'$ as:

$$\begin{aligned} \mathbf{X} &\stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{pmatrix} \\ \mathbf{X}' &\stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{x}'_1 & \boldsymbol{x}'_2 & \cdots & \boldsymbol{x}'_n \end{pmatrix} \end{aligned} \tag{3.1}$$

The exact steps of metric MDS are below.
We define $\boldsymbol{e} \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$. Consider the $n \times n$ centering matrix

$$\mathbf{H} \stackrel{\text{def}}{=} \mathbf{I} - \frac{1}{n} \boldsymbol{e}^{\mathrm{T}} \boldsymbol{e} \tag{3.2}$$

For any $n \times n$ matrix $\mathbf{A}$, it is easy to prove that being left or right multiplied by $\mathbf{H}$ is equivalent to subtracting from each row or column of $\mathbf{A}$ its average respectively. We define the $n \times n$ matrix

$$\mathbf{Z} \stackrel{\text{def}}{=} \begin{pmatrix} \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \cdots & \|\boldsymbol{x}_n\|^2 \end{pmatrix}^{\mathrm{T}} \boldsymbol{e} \tag{3.3}$$

Where $\|\cdot\|^2$ is the length of a vector. From (3.3) we can deduce the matrix algebraic definition of $\mathbf{D}$:

$$\mathbf{D} \stackrel{\text{def}}{=} \mathbf{Z} - 2\mathbf{X}^{\mathrm{T}}\mathbf{X} + \mathbf{Z}^{\mathrm{T}} \tag{3.4}$$

According to (3.3), (3.4) and the centering property and symmetry of $\mathbf{H}$, we define

$$\mathbf{B} \stackrel{\text{def}}{=} -\frac{1}{2}\mathbf{H}\mathbf{D}\mathbf{H} = (\mathbf{X}\mathbf{H})^{\mathrm{T}} (\mathbf{X}\mathbf{H}) \tag{3.5}$$

It is apparent that $\mathbf{B}$ is an inner product matrix. Note that $\mathbf{B}$ is thus positive semidefinite. Therefore, we could obtain another form of $\mathbf{B}$ by singular value decomposition (SVD) and then the closed form of $\mathbf{X}$:

$$\begin{aligned} \mathbf{B} &= \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}} \\ \mathbf{X} &= \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}} \end{aligned} \tag{3.6}$$

Where $\mathbf{U}$ is a $n \times n$ orthogonal matrix, $\mathbf{\Sigma}$ is a diagonal matrix containing all singular values of $\mathbf{B}$, and $\mathbf{\Sigma}^{\frac{1}{2}}$ is the elementwise square root of $\mathbf{\Sigma}$. The ultimate goal of MDS is to reduce the dimensionality of the feature vectors from m to k while maximally preserving the pairwise distances, which could be represented as the following optimization problem:

$$\begin{cases} \arg\min_{\operatorname{rank}\mathbf{X}' \leq k} \left\| \mathbf{B} - \left(\mathbf{X}'\right)^{\mathrm{T}} \mathbf{X}' \right\|_{\mathrm{F}}^2 \\ \sum_{i=1}^{n} \boldsymbol{x}_i = \mathbf{0} \end{cases} \tag{3.7}$$

Where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. According to the low-rank approximation property of SVD, a closed solution of $\mathbf{X}'$ is

$$\mathbf{X}' = \mathbf{V}_k \mathbf{\Sigma}_k^{\frac{1}{2}} \tag{3.8}$$

Where $\mathbf{V}_k$ is the first k rows of $\mathbf{V}$ and $\mathbf{\Sigma}_k^{\frac{1}{2}}$ is a $k \times k$ diagonal matrix formed by the largest k diagonal elements of $\mathbf{\Sigma}$.

Nonmetric MDS does not employ an Euclidean distance matrix, necessitating a solution based on optimization algorithms. The ultimate goal is fundamentally identical to metric MDS, and the problem can be generally framed as the following optimization problem:

$$\begin{cases} \arg\min_{\mathbf{X}'} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2 - d_{ij} \right) \\ \sum_{i=1}^{n} \boldsymbol{x}_i = \mathbf{0} \end{cases} \tag{3.9}$$

An analytical solution typically does not exist, thus requiring numeric calculation. Generally the distance calculation on the solution is Euclidean and the distance matrix $\mathbf{D}$ is still nonnegative, symmetric, and has zeroes as its diagonal elements.

AttentionWalk. AttentionWalk (Abu-El-Haija *et al.*, 2018) is a fast graph embedding algorithm based on random walks, introducing adaptability on the parameters such as random walk step count.

Given a simple graph of n nodes whose adjacency matrix is $\mathbf{A}$. The node are numbered from 1 to n. The edge weights are in the interval [0,1]. The maximal step count is C. We intend to embed the nodes into an m-dimensional vector space and m is even. The embedding is denoted as $\mathbf{Y} \overset{\text{def}}{=} \begin{pmatrix} \mathbf{L} & \mathbf{R}^{\mathrm{T}} \end{pmatrix}$ where $\mathbf{L}$ and $\mathbf{R}$ are $n \times \frac{m}{2}$ and $\frac{m}{2} \times n$ matrices respectively. We define the reconstructed adjacency matrix $\hat{\mathbf{A}} \overset{\text{def}}{=} \mathbf{L}\mathbf{R}$.

An initial configuration $\mathbf{S}$ is provided as an $n \times n$ diagonal matrix, where each diagonal value is the number of random walks starting from the correspondingly num-

bered node. As an algorithm based on random walks, we attempt to discover the expectation matrix $\mathbf{E}$ where Eij equals to the expected count of the random walks from node i to node j. Considering a single step from node u,

$$\mathcal{P}\left(\text{choose node } v \mid \text{currently at node } u\right) = \frac{A_{uv}}{\sum_{w=1}^{n} A_{uw}} \tag{3.10}$$

Random walks are Markov processes. Therefore, according to (3.10), after row normalization $\mathbf{A}$ is converted to a transition matrix $\mathbf{T}$ covering one random walk step. Using the Markov property, we know $\left(\mathbf{T}^k\right)_{ij}$ $(1 \leq k \leq C)$ is the probability of a certain walk being a k-step one from node i to node j.

We define the probability vector $\boldsymbol{q} \overset{\text{def}}{=} \begin{pmatrix} q_1 & q_2 & \cdots & q_C \end{pmatrix}$, where qk represents the probability of a certain walk to have k steps. According to Bayes theorem, the closed form of the probability matrix $\mathbf{P}$ where Pij is the probability of a certain walk being from node i to node j is:

$$\mathbf{P} = \sum_{k=1}^{C} q_k \mathbf{T}^k \tag{3.11}$$

Then it is easy to deduce the closed form of $\mathbf{E}$ with (3.11):

$$\mathbf{E} = \mathbf{SP} = \mathbf{S} \sum_{k=1}^{C} q_k \mathbf{T}^k \tag{3.12}$$

The loss function is similar to cross-entropy loss, respectively treating $\mathbf{E}$ and $\mathbf{I}$ $[\mathbf{A} = 0]$ similarly to the positive and negative labels ina binary classification problem. The final optimization problem is

$$\underset{\mathbf{L},\mathbf{R},\boldsymbol{q}'}{\arg\min} \left\{ - \left\| \mathbf{E} \circ \log \sigma\left(\hat{\mathbf{A}}\right) + \mathrm{I}\left[\mathbf{A} = \mathbf{0}\right] \circ \log\left[1 - \sigma\left(\hat{\mathbf{A}}\right)\right] \right\|_1 + \beta \left\| \boldsymbol{q}' \right\|^2 + \gamma \left\| \hat{\mathbf{A}} \right\|_{\mathrm{F}}^2 \right\} \tag{3.13}$$

Where the probability vector $\boldsymbol{q}$ in (3.12) needed for the closed form of $\mathbf{E}$ is calculated by the parameter $\boldsymbol{q}'$ using the softmax function. $\sigma(\cdot)$ and $\circ$ represent the sigmoid activation function and the elementwise product respectively. $\mathbf{I}\left[\cdot\right]$ is the Iverson notation applied elementwise, yielding 1 when the condition in the given position holds and 0 otherwise. $\|\cdot\|_1$ denotes the 1-norm of a matrix, *i.e.* the mean of the absolute values of all elements. β and γ control the regularization strengths of $\boldsymbol{q}$ and $\hat{\mathbf{A}}$ respectively.

According to (3.12) and (3.13), the closed and differentiable form of the loss function exists. Asa result, the embeddings can be obtained by solving the optimization problem described in (3.13), typically using gradient descent-based methods.

3.4. *Suspicious Pair Filtering*

The suspicious indices for each source code pair can be calculated by the pairwise similarity values of the feature vectors obtained in the previous step. Given two feature vectors $\boldsymbol{x}_1 = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1k} \end{pmatrix}$ and $\boldsymbol{x}_2 = \begin{pmatrix} x_{21} & x_{22} & \cdots & x_{2k} \end{pmatrix}$, their Euclidean distance is

$$\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \tag{3.14}$$

While their cosine similarity is

$$\cos \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle = \frac{\boldsymbol{x}_1 \cdot \boldsymbol{x}_2}{\|\boldsymbol{x}_1\| \|\boldsymbol{x}_2\|} \tag{3.15}$$

Where $\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle$ denotes the angle $\leq \frac{\pi}{2}$ between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Then we filter the source code pairs for the suspicious ones within a certain rank of similarity for manual inspection.

In practical settings, among the suspicious ones, the pairwise string similarity values obtained in the feature extraction step could be another metric to further filter out the source code pairs unnecessary for manual inspection. We can consider within the suspicious ones that also have a string similarity value above a certain low threshold, because actual plagiarizing source code pairs are hardly free of textual similarity.

3.5. *Evaluation*

There are several challenges of evaluating the results of our method. Plagiarizing source code pairs are generally very rare in a submission group if they do exist, and the groups with plagiarizing pairs also tends to be uncommon. The suspiciousness indices are not portable across different methods, rendering the plagiarism state of a pair binary. Determining plagiarism needs manual inspection besides automatic filtering, while it is difficult to ingrain domain knowledge into computers.

The evaluation metric intends to compare the consistency between manual inspection results and the current ones. Therefore, it should consider the plagiarizing pairs alone as they have far more significance and treat them as equal. It is also supposed to be based on the normalized ranks of the suspiciousness indices for a general performance comparison across groups. Ideally it might favor greatly the smaller ranks and penalizes the larger ranks less severely, because smaller ranks indicate larger consistency with manual results and generally larger ranks could greatly fluctuate.

Based on the above criteria, we propose an evaluation metric named geometric mean of normalized ranks (GMNR), which satisfies the above mentioned properties:

$$\text{GMNR} \stackrel{\text{def}}{=} \sqrt[n]{\prod_{i=1}^{n} \frac{r_i}{N}} \tag{3.16}$$

Where N and n are the total and suspicious pair count in a particular group respectively. We consider only the pairs formed by two different source codes that are not removed after data cleaning, and treat the elements of the pairs as exchangeable. r_i $(1 \leq i \leq n)$ are the 1-based ranks of the suspiciousness indices of each plagiarizing pair. It is apparent that a value of GMNR is on $(0,1]$ and a smaller GMNR indicates greater overall consistency with manual results.

4. Experiments

4.1. *Environment*

All experiments are conducted on a computer with 32 GB of RAM. The Python version is 3.9.9. We employ a GTX 2080 Ti for the training and evaluation of the models.

4.2. *Dataset*

We test our method on submissions for the second round of the Certified Software Professional programming contest in 2021. The contest is organized by the China Computer Federation (CCF) and has junior and senior groups that assess programming skills of middle and high school students respectively. It is a onsite contest hold distributedly in provinces. There are two rounds typically in early Octobers and early Novembers, and only contestants passing the first round can participate in the second.

The CSP submission dataset consists of submissions from 25 participating provinces in the second round of CSP 2021. Table 4.1 provides an overview of the dataset. Every participant may submit multiple times for each problem, and only the last submission is

Table 4.1
Overview of the CSP submission dataset

Property	Junior group	Senior group
Problems included	`candy, fruit, network, sort`	`airport, bracket, palin, traffic`
Number of contestants	15073	10644
Number of submissions	52147	35582
Programming language allowed	C, C++	C, C++
Total file size	39.2 MB	44.2 MB

Table 4.2
Parameters of the graph embedding algorithms

Algorithm	Parameter	Value
AttentionWalk	Embedding dimension	512
	Training epochs	50000
	Attention vector length	20
	Walk count	80
	L2 regularization strength	0.01
	Learning rate	5e-5
	GMNR calculation interval	1000
MDS	Dimension of new vector space	Largest power of 2 ≤ group count

rated afterwards and given scores. All types of sensitive information involving personal privacy, such as contestant names and schools, are removed.

We use the method described in Section 3, and conduct experiments with both MDS and AttentionWalk as the graph embedding algorithm. We view all submissions from each province and each task as a submission group, for plagiarism across provinces is practically impossible. For every group, we calculate the minimal GMNR during training only if manual inspection had found any plagiarizing source code pairs, as we intend to compare the results of our method with manual inspection results.

The training parameters for the graph embedding algorithms are in Table 4.2. For AttentionWalk, we use the Adam optimizer. We also apply L2 regularization on the adjacency matrix reconstruction $\hat{\mathbf{A}}$. The GMNR is calculated every 1000 epochs to find the minimal one. For MDS, as the dimension of the embedding vectors cannot exceed that of the original features, adopting an embedding dimension of the largest possible power of 2 provides an adequate trade-off between accuracy and practicality.

4.3. *Results on the CSP Submission Dataset*

All results are even rounded and have four significant digits, unless otherwise noted.

4.3.1. *Junior Group*

Table 4.3 and Table 4.4 show the results on the junior group of the CSP submission dataset using MDS and AttentionWalk respectively. Only provinces of plagiarizing code pairs are shown in tables, and empty cells denote submission groups without known plagiarizing code pairs.

From the tables, our method is robust against different difficulty and skill coverage combinations, and accurately identifies plagiarizing code pairs. On the submission groups of `fruit` our method has an overall lower performance, possibly due to the ability of several ready-made approaches to this problem to obtain a nearly full score, which rarely happens to other problems.

Table 4.3
GMNRs using MDS on the junior group, CSP 2021

Province	candy	fruit	network	sort
Anhui	5.581e-2	3.822e-1	2.378e-1	3.334e-1
Beijing	1.162e-1	2.537e-2		2.978e-2
Guangdong		3.070e-1		
Guangxi	2.167e-1	3.453e-1		2.345e-2
Hunan	3.496e-1			
Jiangsu		2.366e-1		
Sichuan		1.515e-1		
Shandong	3.265e-3	3.190e-1		2.737e-1
Shanghai	1.624e-1			
Shannxi		4.081e-2		
Shanxi	1.589e-1			1.046e-2
Tianjin	3.676e-1	1.975e-1		2.412e-1
Xinjiang				3.928e-1
Yunnan				4.452e-1
Zhejiang	3.128e-1			

Table 4.4
GMNRs using AttentionWalk on the junior group, CSP 2021

Province	candy	fruit	network	sort
Anhui	4.733e-7	5.960e-4	2.394e-4	1.571e-5
Beijing	3.145e-5	7.556e-4		2.011e-5
Guangdong		6.687e-5		
Guangxi	6.656e-6	1.308e-5		1.253e-4
Hunan	1.629e-6			
Jiangsu		3.040e-6		
Sichuan		8.215e-5		
Shandong	6.488e-7	1.387e-4		1.355e-5
Shanghai	5.348e-5			
Shannxi		3.564e-5		
Shanxi	1.919e-5			1.702e-4
Tianjin	7.652e-5	6.724e-4		8.290e-4
Xinjiang				1.057e-3
Yunnan				8.340e-5
Zhejiang	6.567e-6			

4.3.2. *Senior Group*

The results of our algorithm are in Table 4.5 and Table 4.6, with MDS and AttentionWalk respectively. Only provinces of plagiarizing code pairs are shown in tables. Empty cells mean no plagiarism detected in these groups, the same as in Section 4.3.1.

Our method also performs accurately with robustness across all submission groups.

Table 4.5
GMNRs using MDS on the senior group, CSP 2021

Province	airport	bracket	palin	traffic
Chongqing			4.794e-1	
Hubei		1.132e-1		
Jiangsu	5.286e-1			
Jiangxi			1.521e-1	
Sichuan		2.934e-1		
Tianjin		6.330e-2		
Zhejiang	4.120e-1	3.298e-1	4.333e-1	

Table 4.6
GMNRs using AttentionWalk on the senior group, CSP 2021

Province	airport	bracket	palin	traffic
Chongqing			5.018e-5	
Hubei		4.808e-4		
Jiangsu	8.037e-4			
Jiangxi			1.277e-4	
Sichuan		1.508e-5		
Tianjin		4.267e-4		
Zhejiang	4.349e-6	2.858e-6	3.993e-3	

4.4. *Results against Mossad*

We test our method against the Mossad approach[4] to plagiarism detection evasion (Devore-McDonald and Berger, 2020). After applying our method, the rank of the pair of the original and the mutation are consistently below 10 in the 5 groups tested. To the best of our knowledge, our method is the first practical countermeasure against Mossad.

5. Conclusions

We propose an adaptive source code detection method offering robustness and accuracy comparable to conventional methods. We eliminate thresholds in the core parts of our method, easing manual inspection while enhancing adaptability. Real-World tests on the OI dataset indicate the its practicality when faced with the challenges of the varied submission groups and similarity distributions. Almost all known plagiarizing code pairs

[4] Mossad mutates the original submission by inserting repetitive statements and uses `gcc -O3` to determine the semantic equivalence. We insert pre-existing lines instead, as C++ parsing is complex, and choose the first generated mutation with a similarity value by the GST algorithm below 0.4.

have low ranks of suspiciousness index regardless of whether they are syntactically or semantically similar.

Plagiarism detection based on graph embedding can serve as an overlay upon traditional methods, facilitating the transition to adaptive, grey-box algorithms. However, graph embedding lacks sufficient capture of the high-level semantics of the source codes as well as other nuances. More advanced graph algorithms, such as graph neural networks (GNN) might be researched and employed to alleviate this problem.

6. Acknowledgement

This work is supported by State Key Laboratory of Software Development Environment, Beihang University under Grant No. SKLSDE-2022ZX-09.

References

Abu-El-Haija, S., Perozzi, B., Al-Rfou, R.,Alemi, A.A. (2018). Watch your step: Learning node embeddings via graph attention. *Advances in Neural Information Processing Systems*, 31.

Ajmal, O., Missen, M.S., Hashmat, T., Moosa, M., Ali, T. (2013). Eplag: A two layer source code plagiarism detection system. In: *Eighth International Conference on Digital Information Management (ICDIM 2013)* (pp. 256–261).

Arwin, C., Tahaghoghi, S.M. (2006). Plagiarism detection across programming languages. In: *Proceedings of the 29th Australasian Computer Science Conference-Volume 48* (pp. 277–286).

Bandara, U., Wijayarathna, G. (2011).A machine learning based tool for source code plagiarism detection. *International Journal of Machine Learning and Computing* , 1(4), 337.

Cox, M.A., Cox, T.F. (2008). Multidimensional scaling. In: *Handbook of Data Visualization*. Springer, pp. 315–347.

Devore-McDonald, B., Berger, E.D. (2020). Mossad: Defeating software plagiarism detection. *Proceedings of the ACM on Programming Languages*, *4*(OOPSLA), 1–28.

Flores, E., Rosso,P., Moreno, L., Villatoro-Tello, E. (2014). On the detection of source code re-use. In: *Proceedings of the Forum for Information Retrieval Evaluation* (pp. 21–30).

Foltỳnek, T., Meuschke, N., Gipp, B. (2019). Academic plagiarism detection: a systematic literature review. *ACM Computing Surveys (CSUR)*, 52(6), 1–42.

Freire, M., Cebrían, M., Del Rosal, E. (2007). Ac: An integrated source code plagiarism detection environment. *arXiv preprint cs.IT/0703136* .

Gitchell, D., Tran, N. (1999). Sim: a utility for detecting similarity in computer programs. *ACM Sigcse Bulletin*, 31(1), 266–270.

Grover, A., Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 855–864).

Halim, S., Halim, F., Skiena, S.S., Revilla, M.A. (2013). *Competitive Programming 3*. Citeseer.

Jiffriya, M., Jahan, M.A., Ragel, R.G. (2014). Plagiarism detection on electronic text based assignments using vector space model. In: *7th International Conference on Information and Automation for Sustainability* (pp. 1–5).

Karnalim, O., Chivers, W. (2020). Preprocessing for source code similarity detection in introductory programming. In: *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1–10).

Karp, R. M., Rabin, M.O. (1987). Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development* , 31(2), 249–260.

Lee, Y.-J., Lim, J.-S., Ji, J.-H., Cho, H.-G., Woo, G. (2012). Plagiarism detection among source codes using adaptive methods. *KSII Transactions on Internet and Information Systems (TIIS)*, 6(6), 1627–1648.

Ng, A., *et al.* (2011). Sparse autoencoder. *CS294A Lecture notes* , 72(2011), 1–19.

Prechelt, L., Malpohl, G., Philippsen, M. (2000). *Jplag: Finding Plagiarisms among a Set of Programs*. Citeseer.

Rabbani, F.S., Karnalim, O. (2017). Detecting source code plagiarism on. net programming languages using low-level representation and adaptive local alignment. *Journal of Information and Organizational Sciences*, (1), 105–123.

Rahutomo, F., Kitasuka, T., Aritsugi, M. (2012). Semantic cosine similarity. In: *The 7th International Student Conference on Advanced Science and Technology Icast* (Vol. 4, p. 1).

Roberts, E., Camp, T., Culler, D., Isbell, C., Tims, J. (2018). Rising cs enrollments: Meeting the challenges. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 539–540).

Ruff, L., Kauffmann, J.R., Vandermeulen, R.A., Montavon, G., Samek, W., Kloft, M., . . . Müller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*.

Schleimer, S., Wilkerson, D. S., Aiken, A. (2003). Winnowing: local algorithms for document fingerprinting. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (pp. 76–85).

Sulistiani, L., Karnalim, O. (2019). Es-plag: Efficient and sensitive source code plagiarism detection tool for academic environment. *Computer Applications in Engineering Education* , 27(1), 166–182.

Suthaharan, S. (2016). Support vector machine. In: *Machine learning models and algorithms for big data classification* (pp. 207–235). Springer.

Đurić, Z., Gašević, D. (2013). A source code similarity system for plagiarism detection. *The Computer Journal*, 56(1), 70–86.

Wang, D., Cui, P., Zhu, W. (2016). Structural deep network embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1225–1234).

Wise, M.J. (1996). Yap3: Improved detection of similarities in computer program and other texts. In: *Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education* (pp. 130–134).

Yasaswi, J., Kailash, S., Chilupuri, A., Purini, S., Jawahar, C. (2017). Unsupervised learning based approach for plagiarism detection in programming assignments. In: *Proceedings of the 10th Innovations in Software Engineering Conference* (pp. 117–121).

R. Wu has graduated from Beihang University with a bachelor degree in Computer Science and Engineering, and is currently pursuing his Master degree in Beihang. He is involved in the plagiarism detection and technical supporting of the China National Olympiads in Informatics (NOI). His current research interests include computer vision and discrete mathematics.

A. Lv is involved in software development and technical supporting of the China National Olympiads in Informatics (NOI). He graduated from Taiyuan University of Technology with a Bachelor degree in Mathematics, and is currently pursuing his Master degree in Beihang.

Q. Zhao is currently the vice chairman of the scientific committee of the China National Olympiads in Informatics (NOI). He is a lecturer of computer science in Beihang University, working on computer vision and deep learning.

Olympiads in Informatics, 2022, Vol. 16, 107–123
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.09

REPORTS

Primary School Programming Olympiads in Gomel Region (Belarus)

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. The content of programming contests for students in grades 1–4 of the Gomel region (Belarus) is described in this article. A general idea of the thematic content of the tasks and examples of the tasks of the city Olympiad, which took place on April 8, 2016, is provided. The methodology of teaching and preparing junior schoolchildren for such Olympiads is also briefly described. A serious technical basis is the instrumental system of distance learning developed under the control of the author (DL.GSU.BY).

Keywords: programming olympiads, primary school, distance learning instrumental system.

1. Introduction

The popular and difficult task of computer science education is introduction of informatics education in primary school (Dagienė *et al.*,2019). One can see many directions of this work: unplugged education (Plugar, 2021; van der Vegt, 2016), gamification process focused on increase in motivation and engagement of the learners (Combéfis *et al.*, 2016), using Scratch (Fagerlund *et al.*, 2020), performing of certain problem solving tasks of controlling an agent or planning its future behavior – in a digital environment: programmable toy, microworld, programming environment (Kabátová *et al.*, 2016), robot programming (Kanemune *et al.*, 2017), learning visual programming, programming and robotics, and programming and electronics (Panskyi *et al.*,2021), using special software (Tsvetkova *et al*., 2021 and Alemany *et al.*, 2016).

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. This article offers materials for such Olympiads and a brief description of teaching programming and preparing for such Olympiads for students in grades 1–4. Training includes the sequential study of the necessary information and their consolidation by solving the proposed tasks. Verification of solutions is carried out automatically on the DL.GSU.BY website (Dolinsky, 2017).

2. Contents of the Olympiads

Problems for grades 1–4 include three groups of tasks in ascending order of difficulty (each student is invited to solve all these tasks):

Group 1 of tasks (10 tasks): includes tasks from the "Introduction to programming" section: three tasks with numbers (Dolinsky, 2019), one each with symbols, strings, line lengths, position of a character in a line and three tasks for using the built-in programming language Pascal of line processing functions: DELETE, COPY, POS, respectively, delete part of a line, copy part of a line and find the position of the first occurrence of one line into another. In tasks 1–10 (each with 5 points), one needs to write a program that works in accordance with the given examples of input and output:

№1

Output example:
2 0 1 6

№2

Input example:
18
Output example:
t=18 C

Input example:
24
Output example:
t=24 C

№3

Input example:
17 8 1
Output example:
17=8+9 17=1+16

Input example:
21 11 3
Output example:
21=11+10 21=3+18

№4

Input example:
+
Output example:
not +

Input example:
k
Output example:
not k

№5

Input example:
Dog bird mouse
Output example:
s2+s1+s3 bird+dog+mouse

Input example:
mountain sea tree
Output example:
s2+s1+s3 sea+mountain+tree

№6

Input example:
mango snow fabiele
Output example:
(snow)=4 (fabiele)=7 (mango)=5 7-5+4=6

Input example:
honesty forest sophisticated
Output example:
(forest)=6 (sophisticated)=13 (honesty)=7 13-7+6=12

№7

Input example:
Zewitched
Output example:
Bewitched bewitches

Input example:
Kird
Output example:
Bird Birk

№8

Input example:
mou**R**nt**t**ain 4 pin**M**eap**S**ple
Output example:
mountain pineapple

Input example:
gi**p**raf**a**fe 3 va**k**riab**k**le
Output example:
giraffe variable

№9

Input example:
Hor**se** 1 2
Output example:
h se horse

Input example:
mango 3 1
Output example:
man o mango

№10

Input example:
S sunset mouse comicas
Output example:
*unset=1 mou*e=4 omica*=7

Input example:
O mango susurrous honesty
Output example:
mang*=5 susurr*us=7 h*nesty=2

Group 2 of tasks (5 tasks): includes tasks from the "One-dimensional array" section: summing elements, counting elements with a certain property, finding the maximum and minimum elements, and finding the number of an element with a given property. The following are examples of such tasks.

Problem 11 (5 points)

For several weeks Denis saved money that was given to him for pocket expenses. During the autumn holidays, he and his mother decided to go to the park and take a ride on the merry-go-rounds. Calculate the cost of riding on all the carousels and print "+" if Denis has enough money for all the rides, otherwise print the amount of money that you need to ask your mother to have enough to ride on all the carousels.

Input format:	**Input example:**	**Input example:**
S – the amount of money Denis has K – number of carousels (K<=12) a[1] – the cost of riding the 1st carousel a[2] – the cost of riding the 2nd carousel … a[K] – the cost of riding the K-th carousel	1500000 5 20000 15000 12000 17000 25000	50000 4 15000 12000 25000 17000
Output format:	**Output example:**	**Output example:**
s – the cost of riding all carousels + / n – the amount that mom should give	89000 +	69000 19000

Problem 12 (5 points)

Timofey's dad decided to make a basketball ring for his son. The boy has a basket ball with a diameter of 19 cm. To comply with all the rules of basketball, it is necessary that the diameter of the ring is 26 cm larger than the diameter of the ball. Count the number of rings made correctly.

Input format:	**Input example:**
m – number of rings (m<=15) a[1] a[2] … a[m] diameters of rings made by dad	4 45 38 45 41
Output format:	**Output example:**
K – number of rings suitable for Timofey	2

Problem 13 (5 points)

People have always appreciated any kind of thrill-related entertainment. Roller coasters remain the most popular extreme attractions at all times. Calculate the speed of the fastest of the five slides.

Input format:	**Input example:**
a[1] – speed Kingda Ka, km/h a[2] – speed Top Thrill Playster a[3] – speed Formula Rossa a[4] – speed Dodonpa a[5] – speed Steel Dragon	206 190 240 172 120
Output format:	**Output example:**
sp – max speed	240

Problem 14 (5 points)

Spring has come in the land of the Moomins. Baby Sniff picked the first spring flowers for N days. Of the flowers he collected, he gave five to his family members, and from the rest he made bouquets. Calculate the number of flowers in the smallest bouquet that baby Sniff can get.

Input format:	**Input example:**
N – number of days (N<=15) a[1] – number of flowers in the 1st bouquet a[2] – number of flowers in the 2nd bouquet … a[N] – number of flowers in the N-th bouquet	5 7 10 14 9 6
Output format:	**Output example:**
M – number of flowers in the smallest bouquet	1

Problem 15 (5 points)

The Babochkin family is about to go on vacation to the sea. In order to choose the right tour, it is necessary that the day of departure to the sea was the day after the father went on vacation. Determine if there are tour start days that are suitable for the rest of the Babochkin family.

Input format:	**Input example:**	**Input example:**
m – number of tours (m<=10) s – father's day of vacation a[1] – 1st tour day a[2] – 2nd tour day … a[m] – day of the m-th tour	10 11 12 14 15 16 17 13 11 12 14 23	7 29 12 1 6 27 5 18 19
Output format:	**Output example:**	**Output example:**
+ / –	+	–

Group 3 of tasks (5 tasks): designed to differentiate the knowledge of skills and abilities of the most prepared children and includes one simple task on the following topics: two-dimensional array, geometry, strings, research (based on Kangaroo tasks of 2–3 grades), word problem.

Problem 16 (5 points)

In one of the hottest countries in the world, they decided to measure the air temperature for M hours for a week. Determine on which day the maximum temperature was reached and at what time.

Input format:	**Input example:**
M – number of hours to measure temperature (M<=24) a1 a2 … aM – temperature on Monday b1 b2 … bM – temperature on Tuesday c1 c2 … cM – temperature on Wednesday d1 d2 … dM – temperature on Thursday e1 e2 … eM – temperature on Friday f1 f2 … fM – temperature on Saturday g1 g2 … gM – temperature on Monday	5 28 28 29 29 30 30 31 31 31 31 32 32 32 32 32 30 30 30 31 31 32 33 33 33 34 31 31 32 33 33 29 29 30 31 31
Output format:	**Output example:**
max – highest temperature n – the hour when the maximum temperature was reached	34 5

Problem 17 (5 points)

Pavel is going to visit all excursions during L days of rest at sea. Determine the farthest excursion from the place of residence, and how far he will travel in all days if he visits one excursion every day and returns to the hotel.

Input format:	**Input example:**
L – number of rest days (L<=14) x1 y1 – coordinates of the 1st excursion x2 y2 – coordinates of the 2nd excursion … xL yL – coordinates of the L-th excursion xe ye – coordinates of the place of residence	7 1 1 4 2 7 1 1 5 8 10 12 6 6 6 6 5
Output format:	**Output example:**
m – distance to the furthest excursion nom – the number of the furthest excursion rast – total distance Display real numbers with 1 decimal place.	6.4 1 63.2

Problem 18 (5 points)

N lines are given containing characters '.' and '#'. Print the number of the first line containing the least number of '#' characters.

Input format:	Input example:
N (N<=5) s1 s2 .. sN	4 .##.. .... ###### ##.####
Output format:	**Output example:**
k - line number	2

Problem 19 (5 points)

The difference between the two numbers is X less than the number to be subtracted and by Y more than the subtracted one. What is it equal to?

Input format:	Input example:
X Y	3 4
Output format:	**Output example:**
Z - difference of two numbers	7

Problem 20 (5 points)

Input file: input.txt Output file: output.txt

While e-books are gaining popularity around the world, in Byteland, everyone loves to read books in the library.

The National Library of Byteland has a variety of shelves, each containing books on a specific topic. The most popular shelving unit is a selection of fiction. The rack has **N** shelves, each of which holds **N** books.

Sometimes readers return a book to the shelf and put it on the wrong shelf. Therefore, every evening the librarian Eleonora Romualdovna puts things in order in the rack. In total, the National Library of Bytelandia contains works of art by **N** different authors, and the library contains $\mathbf{a}_i$ books of the **i**-th author. Eleonora Romualdovna defines the disorder on the shelf by the number **P**, which is equal to the maximum of $\mathbf{p}_i$ values, where $\mathbf{p}_i$ is the disorder on the i-th shelf. The clutter on the **i**-th shelf is calculated as the number of different authors whose works are located on it. Eleonora Romualdovna believes that the rack is in perfect order if the number **P** is minimal.

Arranging books every day, deciding where to put each book, is a very difficult task, so Eleonora Romualdovna asks you to help. To begin with: count how many books she has in the library.

Input data:

The first line of the input file contains a single natural number **N** ($1 \le \mathbf{N} \le 100$) – the number of shelves.

The second line contains **N** integers $\mathbf{a_i}$ ($1 \le \mathbf{a_i} \le 10^5$) – the number of books by the **i**-th author.

The numbers in the lines of the input file are separated by single spaces.

Output data:

The output file should contain one number – the total number of books in the library.

input.txt	*output.txt*
3 1 2 6	9

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 a school year: in October-November for school and city at three divisions: grades 1–4, 5–8, 9–11 and in March–April for school, city and regional at three divisions: grades 1–4, 5–7, 8–9. When conducting these Olympiads, Internet technologies and the DL.GSU.BY website are used, which allows not only schoolchildren of the Gomel region to participate in all the Olympiads, but also everyone who wishes. And, it should be noted, there are dozens of such applicants from all regions of Belarus and the city of Minsk.

3. Training 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 subsequently choose information technology as their professional field, and for everyone who will be engaged in at least some time. Practice also shows that the training is structured in a rather interesting form. All classes are conducted only on a voluntary basis outside the classroom. Another equally important aspect is the differentiated approach. The use of Internet technologies makes it possible to provide individual training along a personal educational trajectory.

The work with first-graders begins with the course “Learning to think”(Dolinsky, 2014). A side effect of the lessons in this course is the growing interest in teaching a wide range of younger students. The main goal is to acquire stable skills in performing basic mental operations. At the moment, the course offers the following basic mental operations (in the amount of 21 pieces):

- Operations on pairs: comparison, ordering, association.
- Operations on sets: union, intersection, subtraction.
- Operations on a set: classification, structuring, generalization.
- Logical operations: negation, conjunction, disjunction, equivalence, implication.

Complex operations: synthesis, memorization, analysis, imagination, analogy, abstraction, positioning.

Further training is consistently conducted within the following sets of tasks “Introduction to programming”, “Debugger”, “One-dimensional array”, “Two-dimensional array”, “Geometry”, “Strings” (Dolinsky, 2013). Learning in all these sets of tasks is built on the principles of differentiated learning (Dolinsky, 2020). Stem tasks are listed in ascending order of difficulty. For each stem problem, there is a branching tree of leading problems of less complexity. In the end, for each task, training is provided with the presented source code in approximately the following order.

Each studied task after a certain number of tasks is met as a control one. In this case, there is no “Don’t know” button. If a student cannot solve such a problem (previously studied) even with the help of his notebook, he is automatically transferred back to learning to this problem. To stimulate more intense thinking activity, as opposed to thoughtlessly pressing the “Don’t Know” buttons, most of the lead-in folders are provided with some of the test assignments described above, BEFORE and AFTER learning to solve the problem for which the student pressed the “Don’t Know” button.

Folders with tasks for the development of basic mental operations are continuously interwoven into the learning process, both on the basis of graphic images and on the basis of using the studied material as graphic images in the form of tests, algorithms and programs.

For a more complete control of the assimilation of topics, at the end of each of them there are folders with analogy problems. If a student finds it difficult to solve them, then it is necessary to improve learning in general and work with this student, in particular.

Next, there are tasks, for the solution of which the ability to combine the studied methods of solving problems is required.

Finally, each topic ends with a complete set of available Olympiad problems. Since April 2007, programming contests have been held in the Gomel region for students in grades 1–4. First, there are olympiads problems on a given topic, and then sets of olympiads problems on all topics studied up to this topic, inclusive.

4. Conclusion

The materials of programming contests for primary school students and briefly presents the methodology for teaching and preparing junior schoolchildren for such Olympiads are considered in this article. The Olympiad for pupils of grades 1–4 of Gomel and the Gomel region, held on April 8, 2016, was attended by 50 students from 10 settlements: Gomel, Rechitsa, Zhlobin, Kalinkovichi, Mozyr, Svetlogorsk, Chechersk (all-Gomel region), Grodno, Lida (Grodno region), Polotsk (Vitebsk region). The winner (4th grade student Kopichenko S.) solved all the proposed problems. Diplomas were awarded to three 1st grade students, two 2nd grade students, five 3rd grade students and eight 4th grade students from Gomel, Zhlobin, Svetlogorsk and Kalinkovichi. The wide geography and high results confirm both the correct choice of a set of tasks and the effectiveness of the proposed distance learning system. The author’s solutions to the problems of this Olympiad are attached to the article.

References

Alemany 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, Characterisation and Discussion. *Olympiads in Informatics*, 10, 39–60.

Dagienė V., Jevsikova T., Stupurienė G. (2019) Introducing Informatics in Primary Education: Curriculum and Teachers' Perspectives. In:

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 Informatic. *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, 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`, 14, 37–46.

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

Performance Statistics of Gomel pupils at international and national olympiads in informatics since 1997 up to 2020 (In Russian): `http://dl.gsu.by/olymp/result.asp`

Plugar Z. (2021). Extending Computational Thinking Activities. *Olympiads in Informatics*, 15, 83–89.

Pozdniakov S., Dagienė V. (eds) *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 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.

Appendix.

The following are the source codes for solving presented problems in Pascal:

Problem 1

```
begin
  writeln(2);
  writeln(0);
  writeln(1,' ',6);
end.
```

Problem 2

```
var
  a : longint;
begin
  readln(a);
  writeln('t=',a,' C');
end.
```

Problem 3

```
var
  s1,s2,s3,s4,s5 : longint;
begin
  readln(s1);
  readln(s2,s3);
  s4:=s1-s2;
  s5:=s1-s3;
  writeln(s1,'=',s2,'+',s4);
  writeln(s1,'=',s3,'+',s5);
end.
```

Problem 4

```
var
  a : char;
begin
  readln(a);
  writeln('not ',a);
end.
```

Problem 5

```
var
  a,b,c : string;
begin
  readln(a);
  readln(b);
  readln(c);
  writeln('s2+s1+s3');
  writeln(b,'+',a,'+',c);
end.
```

Problem 6

```
var
  s1,s2,s3 : string;
  d1,d2,d3 : longint;
begin
  readln(s1);
  readln(s2);
  readln(s3);
  d1:=length(s1);
  d2:=length(s2);
  d3:=length(s3);
  writeln('(',s2,')=',d2);
  writeln('(',s3,')=',d3);
  writeln('(',s1,')=',d1);
  writeln(d3,'-',d1,
          '+',d2,'=',d3-
d1+d2);
end.
```

Problem 7

```
var
  s : string;
  c : char;
begin
  readln(s);
  c:=s[1];
  s[1]:='b';
  writeln(s);
  s[length(s)]:=c;
  writeln(s);
end.
```

Problem 8

```
var
  s,p : string;
  k,d : longint;
begin
  readln(s);
  readln(k);
  readln(p);
  d:=length(s);
  delete(s,d-k+1,1);
  delete(s,k,1);
  d:=length(p);
  delete(p,d-k+1,1);
  delete(p,k,1);
  writeln(s);
  writeln(p);
end.
```

Problem 9

```
var
  s,p,q   : string;
  d,k1,k2 : longint;
begin
  readln(s);
  readln(k1,k2);
  d:=length(s);
  p:=copy(s,1,k1);
  q:=copy(s,d-k2+1,k2);
  writeln(p,' ',q,' ',s);
end.
```

Problem 10

```
var
  s1,s2,s3 : string;
  c        : char;
  p1,p2,p3 : longint;
begin
  readln(c);
  readln(s1);
  readln(s2);
  readln(s3);
  p1:=pos(c,s1);
  p2:=pos(c,s2);
  p3:=pos(c,s3);
  s1[p1]:='*';
  s2[p2]:='*';
  s3[p3]:='*';
  writeln(s1,'=',p1);
  writeln(s2,'=',p2);
  writeln(s3,'=',p3);
end.
```

Problem 11

```
var
  a       : array [1..12] of longint;
  s,i,n,p : longint;
begin
  readln(p);
  readln(n);
  for i:=1 to n do readln(a[i]);
  s:=0;
  for i:=1 to n do s:=s+a[i];
  writeln(s);
  if s<=p
    then writeln('+')
    else writeln(s-p);
end.
```

Problem 12

```
var
  a     : array [1..15] of longint;
  k,i,n : longint;
begin
  readln(n);
  for i:=1 to n do read(a[i]);
  k:=0;
  for i:=1 to n do
    if a[i]=45 then k:=k+1;
  writeln(k);
end.
```

Problem 13

```
var
  a   : array [1..5] of longint;
  k,i : longint;
begin
  for i:=1 to 5 do readln(a[i]);
  k:=a[1];
  for i:=2 to 5 do
    if a[i]>k then k:=a[i];
  writeln(k);
end.
```

Problem 14

```
var
  a     : array [1..15] of longint;
  i,k,n : longint;
begin
  readln(n);
  for i:=1 to n do readln(a[i]);
  k:=a[1];
  for i:=2 to n do
    if a[i]<k then k:=a[i];
  writeln(k-5);
end.
```

Problem 15

```
var
  a     : array [1..10] of longint;
  k,i,s : longint;
begin
  readln(k);
  readln(s);
  for i:=1 to k do readln(a[i]);
  i:=1;
  while (i<=k) and (a[i]<>s+1) do i:=i+1;
  if i>k
    then writeln('-')
    else writeln('+');
end.
```

Problem 16

```
var
  i,j,n,m,a,max : longint;
begin
  readln(m);
  max:=-maxlongint;
  for i:=1 to 7 do
    for j:=1 to M do
      begin
        read(a);
        if a>max
          then begin max:=a; n:=j; end;
      end;
  writeln(max);
  writeln(n);
end.
```

Problem 17

```
var
  x,y            : array [1..14] of real;
  xe,ye,m,rast,d : real;
  i,L,nom        : longint;
begin
  readln(L);
  for i:=1 to L do readln(x[i],y[i]);
  readln(xe,ye);
  rast:=0; m:=0;
  for i:=1 to L do
    begin
      d:=sqrt(sqr(xe-x[i])+
              sqr(ye-y[i]));
      rast:=rast+d;
      if d>m
        then begin m:=d; nom:=i; end;
    end;
  writeln(m:0:1);
  writeln(nom);
  writeln(2*rast:0:1);
end.
```

Problem 18

```
var
  i,j,k,min,n,nom : longint;
  s               : string;
begin
  min:=maxlongint;
  readln(n);
  for i:=1 to n do
    begin
      readln(s);
      k:=0;
      for j:=1 to length(s) do
        if s[j]='#' then inc(k);
      if k<min
        then begin min:=k; nom:=i; end;
    end;
  writeln(nom);
end.
```

Problem 19

```
var
  x,y : longint;
begin
  readln(x,y);
  writeln(x+y);
end.
```

Problem 20

```
var
  a     : array [1..100] of longint;
  i,N,K : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(N);
  for i:=1 to N do read(a[i]);
  k:=0;
  for i:=1 to N do inc(k,a[i]);
  writeln(k);
  close(input); close(output);
end.
```

Olympiads in Informatics, 2022, Vol. 16, 125–133
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.10

Olympiads in Informatics in Kyrgyzstan

Pavel S. PANKOV[1], Kylychbek A. URAIYMOV[2],
Artem A. BELYAEV[3]

[1]*Institute of Mathematics, Kyrgyzstan*
[2]*International Educational Institution "Sapat", Kyrgyzstan*
[3]*Kyrgyz-Russian Slavic University, Kyrgyzstan*
e-mail: pps5050@mail.ru, kylychbek.uraimov@sapat.edu.kg, artem_belyaev@mail.ru

Abstract. This report provides description of the Multi-stage National Olympiad along with other inner competitions in informatics, the online platform to conduct competitions in informatics in Kyrgyzstan, the participation in IOI and in other international olympiads in informatics.

This report also presents the principles and ways to create tasks (naturalness; presentation of real processes; reference to known objects; non-Euclidean spaces; "brute force method is either inapplicable or gives too overestimate of complexity") and examples of tasks of various levels with classification.

Keywords: Kyrgyzstan, National Olympiad, informatics, task, site.

1. Introduction

The conduct of Olympiads in informatics in Kyrgyzstan since 1985 was described (Pankov *et al.*, 2007; Pankov *et al.*, 2011).

- **Section 2** contains the description of the multi-stage National Olympiad, other competitions in informatics in Kyrgyzstan, participation in IOI and other international olympiads in informatics.
- **Section 3** describes the online platform to conduct competitions in informatics in Kyrgyzstan.
- **Section 4** presents a survey of principles and ways to create tasks (naturalness; presentation of real processes; reference to known objects; non-Euclidean spaces; "brute force method is either inapplicable or gives too overestimate of complexity").
- **Section 5** contains examples of tasks of various levels created with methods of Section 4.

2. Competitions

Teaching informatics (under the traditional name "Foundations of Informatics and Computer Facilities") in secondary schools of Kyrgyzstan started in 1985.

The Olympiads in Bishkek city, the capital of Kyrgyzstan, are conducted since 1985 (annually in January).

Republican Olympiads in informatics, as a constituent of Republican Olympiads in various subjects of secondary schools are conducted since 1987.

Stage I is schools' (November); stage II is rayons' (December); stage III is: 7 regions, Bishkek city and Osh city (February or March); stage IV is final (March).

(Now Republican Olympiad in informatics is a constituent of National Olympiad).

Selection competitions to IOI for prize-winners of IV stage (April) were conducted in English since 2000 until 2019.

National Olympiads in informatics, as a constituent of selection to International Olympiads (IOI, IMO, IPhO, IChO, IBO) are conducted since 2020.

Stage I is for all comers, online (January); stage II is online (February); stage III is on-site (March); stage IV (selection to IOI) is for prize-winners of stage III and of the final stage of Republican Olympiad, on-site (April).

Kyrgyzstan participates in IOI since 2000. Our achievements are bronze medals at IOI'2000, IOI'2004, IOI'2005, IOI'2016, IOI'2017, three bronze medals at IOI'2019, four bronze medals at IOI'2021.

Every year we conduct ICPC Kyrgyzstan Championship for students as a quarter-final of International Collegiate Programming Contest (November). Schoolchildren's teams participate in it out of competition.

Since 2015 our teams participate in International Zhautykov Olympiads (IZhO), our achievements in "Computer science" nomination are: 2015: 3 bronze; 2016: silver; 2017: gold, silver, bronze; 2018: silver, 2 bronze; 2019: 3 bronze; 2020: 2 bronze; 2021: 5 bronze; 2022: 2 gold, 4 bronze.

Since 2015 our teams participate in Asia-Pacific Informatics Olympiads (APIO), our achievements are: 2017: bronze; 2018: 3 bronze; 2019: silver; 2020: bronze; 2021: 2 bronze.

European Junior Olympiad in Informatics (EJOI), 2021: gold.

XIII Eurasian Olympiad in Informatics, Central Asia, 2021: first place.

3. The Online Platform

All competitions in informatics are conducted on the base of the site:

`olymp.krsu.edu.kg`

The site supports providing competitions of various types: according to the ICPC rules or student competitions; when a complete solution of the task is required or a partial solution is possible.

The system provides the following features:

Possibility to judge the solutions on different physical computers; automatic registration of the competitors; separation of access privilege to the contests; support of any programming languages (C++, C#, Java, Python, Pascal etc.) and the ability to add new languages easy setup without changing the application code; automatic verification solutions for malicious code; automatic re-checking solutions by the administrator's signal; limited allocation of computing resources (CPU time, random-access store, external memory) for solutions; statistical analysis of the problems solving process; virtual competitions support.

Architecture Components: Dispatcher (distribution of solutions between judge services), Judges (checking solution; compilers management); Web application and database.

Hence, the system has two components:

1. The site or web application itself, through which the user interacts with the system.
2. Dispatcher and judges who check and evaluate solutions.

Since these components are independent, they can be scaled. Solution verification can be run on completely different machines, which provides flexibly manage_of the load during various competitions. The task format is compatible with the `polygon.codeforces.com` system. It provides preparing problems and using ready-made problems without additional modifications.

4. Principles and Ways to Develop Tasks

We try to prepare tasks to be well-understood, to have "short and elegant formulations" (Dagienė *et al.*, 2007).

The following items are not defined exactly; they are overlapping somewhere. Certainly, we do not pretend to originality. We use such papers as (Burton *et al.*, 2008), (Diks *et al.*, 2008), (Kemkes *et al*., 2007) and

4.1. Naturalness (Pankov, 2008) includes: presentation of real processes (Pankov, 2010) with gravitation, repelling and attracting; a human can „see" the answer for corresponding task image with initial data of small volume without calculations; tasks are difficult to be solved even with initial data of small volume; a „brute force" method is either inapplicable or gives too overestimation of complexity. Sometimes it is not necessary to write a program for generation of tests for a task. A human can compose sufficiently complex tests where the answer is "seen" but it is difficult to find it by means of any program.

4.2. Reference to known objects (Pankov *et al.*, 2009) including local circumstances, the host town, the host state, sponsors (at the same time, the task should be "culturally neutral").

4.3. For solving an Olympiad task by the contestant, in addition to the common limits in the CPU time (traditionally 1 second) and in memory (traditionally 256 megabyte) there exists an actual limit (*) on average time to write and

debug a program even if the contestant has necessary skills and vision of the algorithm (about 1–1.5 hours). Sometimes such limit is achieved by means of a long-winded and complicated text of the task, with many "permissions" and "bans". In the proposed tasks this limit is achieved naturally, because of their geometrical content.

4.4. Unusual spaces in tasks are built sometimes by null-transportation etc. We propose to use "natural" non-Euclidean spaces: Moebius band, Riemann surfaces, topological torus, projective plane.

4.5. Using "regular" graphs instead of "general" graphs involves very vast graphs with short and well-understood description, for instance (combination with 4.4): Take a 2022 × 2022 square grid and add arcs: (1,1)-(2022,2022), (1,2)-(2022,2021), … (1,2022)-(2022,1) (Moebius band).

4.6. More than one actor or non-point actor within a graph.

4.7. Pseudo-game. How many moves are necessary to defeat an actor operating by a known (declared in the text of the task) algorithm? Attempts to write such a program are more effective than a "logical" way to solve the task.

4.8. Tasks of a priori unbounded complexity including search in infinite spaces (Pankov *et al.*, 2012; Pankov *et al.*, 2018). The contestant is to reduce the task to search in finite space logically.

4.9. Guessing theorems by the contestant (particularly, for reducing in 4.8). While creation of the task the jury is to prove such theorems strictly but the contestant uses them swiftly. Catching sight of regularities in beginnings of sequences also is effective.

4.10. Discrete tasks of continuous content (Pankov, 2013). They are difficult to solve because the optimal way is mixed: to grope the optimal solution by means of Analysis and to find it by "brute force in small domain".

4.11. Pattern recognition (with strict formulation) (Pankov *et al.*, 2020).

4.12. Real processes executed by 2D- and 3D-printers (with nouns of pixels, voxels, spexels, timexels: space primitives existing during one temporal step) and hypothetical 4D = (3D + time)-printers (Pankov *et al.*, 2021).

4.13. Geometrical tasks for pixels. They have the following preference: numbers of pixels ("area" of any figure or "length" of a "segment") are defined and calculated directly.

4.14. Turkic languages are agglutinating and have strict rules to add affixes. For example, consequence of vowels can be AYAY… or EIEI… or OUA… only; KITEP(book) + DA(locative) + BY(?) = KITEPTEBI(in the book?).
It gives capacities to create tasks on lexicographical order, on numbers of "words" meeting some rules.

4.15. Non-substantiated but practically effective methods (can be used by the contestant if results are seen during the content or the contestant risks). If all or many tests are passed then the jury will not check the text of program. For instance, greedy search with some improvements.

4.16. We do not mean and propose any general algorithms to solve some tasks. On the contrary, we suppose that such algorithms with traditional estimation *O(N...)* in some cases (4.1, 4.3, 4.8, 4.10, 4.11) do not exist and preferences of

such tasks are that each task demands its own algorithm, with little discoveries, to smooth out the effect of training contestants.

4.17. To avoid any cribbing, we use parameterized tasks sometimes (Pankov *et al.*, 2015).

5. Examples of Tasks

We cite tasks of various levels, since 2002. Some of tasks are cited non-formally, not completely. We omit the ranges of input and scoring. Also, we use "general tasks" which can be specified according to the level of competition.

We write "input" in examples in abbreviated form: lines are separated with the sign \ .

Task 1 (Horse). Let Lake looks like an isosceles triangle, the basis of the triangle (northern coast) is 190 km and height (width of Lake) is 60 km. Village is located on northern coast of Lake at 20 km from the western corner. Horse runs with speed of 20 km/hour and swims with speed of 10 km/hour. Write a program: A) to show Lake and Village; B) to enable User to show any point on the coast of Lake; C) to draw the fastest way for Horse or D) to show the motion (in scale of 1 hour = 1 sec.) of Horse from this point up to Village along such way.

Comments. 2002 was the year of Horse. The task reflects a historic fact. This village was named after Horse which had crossed the Issyk-Kul lake in XVIII century.

Task 2 (Mice). Given a graph, its vertices are "houses". The Instrument has counted mice under each of houses at different moments. During all this measuring, each mouse could pass to another neighbor house only once. Write a program to find the least possible number of all mice.

Example. Six houses form a ring. Input: 9, 0, 1, 0, 0, 2. Output: 10. [Two mice under the first house and two mice under the sixth one could be the same].

Task 3 (Train). A graph is given. Firstly, the head *H* and the tail *T* of a train are in two neighbor vertices. Write a program finding one of the shortest ways to be passed by the train (moving forward only) in order to put its head to the primary position of *T* and its tail to one of *H*.

Task 4 (Snow). Let the streets in the city form a rectangular grid. The firm *Logic* [sponsor] is situated 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 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.

Task 5 (Mouse). At night, a mouse is anywhere within a long ditch of "figure-of-eight" of length 2008 meters, the first ring of the ditch is numbered from 0 till 1004 (from the cross to the cross) and the second ring is numbered from 1004 till 2008 (the points with

numbers 0, 1004 and 2008 coincide). The mouse can run quickly but cannot climb out. Two men with sacks stand at *X1* and *X2* meters. The men's velocity is 1 meter/second. Write a program calculating the minimal time to catch the mouse in any case.

The following three tasks are specifications of the general task on grammar of Turkic languages.

Task 6 (Vowels). "Words" contain vowels A, E, I, O, U, Y. There must be either consecutive same vowels or the following pairs of consecutive different vowels: AY, YA, EI, IE, OU, UA. Given is a "word" *W* containing more than one vowel. At least how many vowels must be erased from *W* to obtain a new word, the sub-sequence of vowels of which contains only permitted pairs of consecutive vowels?

Example. Input: TOOFEIGUZAEEWYQ Output: 4

Task 7 (Vowels-2). How many words of given length L, made of letters C, A, E, I, O, U, Y having at last one vowel and meeting conditions of Task 6 exist? Output (this number mod 1000).

Task 8 (Vowels-3). Given a word meeting conditions of Task 7. What is its number in lexicographical sequence of all such words with same length? Output (this number mod 1000).

Example. Input: AY Output: 2

Task 9 (3D-printer). The *X*- and *Y*-axes are horizontal, the *Z*-axis is down. Sides of all cubes are equal 1 and parallel to the axes, coordinates of their centers are integer numbers. The lower semi-space "$Z \le 1$" is filled with cubes. The initial coordinates of Cube-printer are (0, 0, 0). Given one, two or three cubes with coordinates in $[-N, N] \times [-N, N] \times [1, N]$. At each step Cube-printer moves by one along one of axes and erases a met cube. How many steps of Cube-printer are necessary to erase the given cube(s) (with cube(s) over them only)?

Example for two cubes: Input: 2 \ 8 7 1 \ 9 7 5 Output: 21

It is seen that the complexity of this task does not depend on $N > 10$.

Task 10 (Robot). Cubic Robot of volume 1 moves in continuous media (for instance, the warm iron cube moves in dense snow). Its edges are parallel to *X*-, *Y*-, *Z*-axes. A "shift" of Robot is its motion along or against one of the axes by an integer number *J* ($|J| \le 10^{12}$). Given a sequence of 2..6 shifts, find the volume of Robot's "trace" (empty space in media made by Robot's motions including Robot itself).

Example for three shifts. Input: 3 \ Z 5 \ X –6 \ X 4 Output: 12

Task 11 (Triangle). All numbers are integer; square grid is considered.

Given two points A and B different from the origin of coordinates O (four numbers XA, YA, XB, YB in -1000000..1000000). At each step point B can move along a side or along the diagonal of a cell. To obtain a rectangular triangle OAB how many steps are necessary?

Example: Input: 900000 0 900500 0 Output: 500

Task 12 (Virus). Given the initial position of Virus (integer numbers XV, YV) on an N×N-square grid. At each step you can put an obstacle on a point of grid. In response, Virus tries to step to the neighbor point consequently East; North; West; South. How many steps are necessary to catch Virus?

Example: Input: 1 1 Output: 4

It is seen that the complexity of this task does not depend on $N > 10$.

Task 13 (Sulaiman-mount) [with Museum, in the middle of city of Osh]. Mice are going to celebrate New Year – Spring Equinox on Top of Mount. Now there are M mice and J nuts at Museum and T nuts at Foot of Mount. Time of Foot-Museum movement and Museum-Top one is 10 minutes. One mouse can carry one nut. Find the minimum time (minutes) to deliver all nuts to Top.

Example: Input: M = 3, J = 2, T = 2 Output: 50

General Task 14 (Cell connection). Given some distinct points on a square grid. What is the minimal total length of broken line(s) drawn along sides of cells and connecting all given points?

[Using geometrical proximity yields more effective algorithm than ones for general graph].

By our experience, some who have good command over programming do not know mathematics. Every year we give

General Task 15 (Geometry). Given two figures (rectangles, triangles, segments …) with integer coordinates of vertices (of endpoints …). Find the area of the intersection (of a figure generated by these figures …). Output: P/Q

where P and Q are natural numbers and GCF(P,Q) = 1 (Q may be 1).

There "naturally" arise many special cases in such tasks and contestants spend much time.

Task 16 (Pixels). At first, in an N×N-display boundary pixels are yellow and other pixels are red. If the pixel (X Y) is red then by this command itself and pixels "up, down, left, right" consequently are painted yellow. By the command "W" painting is run until the boundary along each of four directions; by the command "F" painting is run until the first met yellow pixel along each of for directions. After executing of 2..10 such commands how many red domains will turn out?

Example for three commands: Input: 3 \ W 2 4 \ F 3 5 \ W 4 3 Output: 3

It is seen that the complexity of this task does not depend on $N > 100000$.

Task 17 (Roads). Denote points Bishkek(B), Suusamyr Fork(A), Jalal-Abad(J), Talas(T), Osh(O), Karakol(K), Balykchy(L), Naryn(N), Batken(E). Highway distances are: AT = 102; AJ = 377; AB = 193; JO = 106; OE = 240; LB = 179; LK = 216; LN = 180 (km). The speed limit is 60 km/hour. Competition is announced for far apart pairs of drivers. Because of Covid-19, distance (along highway) between them must not be less than 5 km. Now the first driver with car is at X1 point and the second driver with car is at X2 point. The first driver is to reach Y1 point and the second driver is to reach Y2 point ($X1 \neq X2$, $Y1 \neq Y2$), $(X1, X2) \neq (Y1, Y2)$). Find the minimal time (minutes) for it.

Example: Input: J O O J Output: 971

Task 18 (Rectangles). An image is presented at an (white) N × N-display by black pixels. Split the image into the minimal number of (non-overlapping) rectangles. Output this number.

Example ($N = 4$): Input: 0001 \ 0110 \ 1111 \ 0110 Output: 4

Task 19 (Rectangles-2). … Present the image as the union of minimal number of rectangles. Output this number.

Example ($N = 4$): Input: (the same) Output: 3

Task 20 (Embeddings). A natural number N and two words of lengths more than 2 are given. Output a word of length N containing these words as many times as possible, and number of these "embeddings".

If there are some such words then output the first of them in lexicographical order. If there are not such word then output NO 0

Example 1: Input: 8 NZG ZNZ Output: ZNZGZNZG 4 [NZG does 2 times, ZNZ does 2 times]

Example 2: Input: 9 LLL BKTL Output: LLLLLLLLL 7 [LLL does 7 times, BKTL does 0 times]

6. Conclusion

We hope that this paper would diversify the scope of tasks for informatics olympiads, making them more engaging for young people, and attracting contestants' attention to vast applications of informatics, inspire a greater interest of young people in learning sciences and perhaps even in helping to make an appropriate career choice in future. Also, we invite to make acquaintance with tasks of our preceding Olympiads at

`https://olymp.krsu.edu.kg/GeneralProblemset.aspx`

References

Dagienė, V., Skupienė, J. (2007). Contests in programming: quarter century of Lithuanian experience. *Olympiads in Informatics: Country Experiences and Developments*, 1**,** 37–49.

Pankov, P.S., Oruskulov, T.R. (2007). Tasks at Kyrgyzstani Olympiads in Informatics: Experience and Proposals. *Olympiads in Informatics: Country Experiences and Developments*, 1**,** 131–140.

Pankov, P.S. (2008). Naturalness in Tasks for Olympiads in Informatics. *Olympiads in Informatics: Country Experiences and Developments*, 2**,** 16–23.

Pankov, P.S., Baryshnikov, K.A. (2009). Representational Means for Tasks in Informatics. *Olympiads in Informatics*, 3**,** 101–111.

Pankov, P.S. (2010). Real Processes as Sources for Tasks in Informatics. *Olympiads in Informatics*, 4**,** 95–103.

Pankov, P.S., Oruskulov, T.R. (2011) Kyrgyzstan National Report on Olympiads in Informatics. *Olympiads in Informatics,* 5**,** 155–160.

Pankov, P.S., Baryshnikov, K.A. (2011). Tasks of "Mission impossible" and "Mission impeded" types. *Olympiads in Informatics,* 5**,** 113–119.

Pankov, P.S., Baryshnikov, K.A. (2012). Tasks of a Priori Unbounded Complexity. *Olympiads in Informatics*, 6**,** 110–114.

Pankov, P.S. (2013). Tasks in Informatics of Continuous Content. *Olympiads in Informatics*, 7**,** 101–112.

Pankov, P.S., Baryshnikov, K.A. (2014). Tasks in Informatics with Pre-Existing Algorithms. *Olympiads in Informatics*, 8**,** 145–155.
Pankov, P.S., Janalieva, J.R. (2015). Conducting complex competitions in informatics with individual tasks. *Olympiads in Informatics*, 9**,** 163–172.
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., Kenzhaliev, A.A. (2020).. Pattern Recognition and Related Topics of Olympiad tasks. *Olympiads in Informatics,* 14, 143–150.
Pankov, P.S., Imanaliev, T.M., Kenzhaliev, A.A. (2021). Automatic Makers as a Source for Olympiad Tasks. *Olympiads in Informatics*, 2021, 15**,** 75–82.
Burton, B. A. Heron, M. (2008). Creating Informatics Olympiad Tasks: Exploring the Black Art. *Olympiads in Informatics: Tasks and Training*, 2, 16–36.
Diks, K., Kubica, M., Radoszewski, J., Stencel, K. (2008). A proposal for a task preparation process. *Olympiads in Informatics: Tasks and Training*, 2**,** 64–74.
Kemkes, G., Cormack, G., Munro, I., Vasiga, T. (2007). New task types at the Canadian computing competition. *Olympiads in Informatics: Country Experiences and Developments*, 1**,** 79–89.

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, 2018–2021. Graduated from the Kyrgyz State University in 1969, is a head of laboratory of Institute of mathematics of KR NAS.

K.A. Uraiymov (1989), International Educational Institution "Sapat". Deputy leader at IOIs, 2019–2021. Coach of 9 medalists at IOI, coach of over the 100 medalists of any informatics Olympiads and coach of all winners of National Olympiads in informatics since 2014 until present. Graduated from the Kyrgyz Tukish "Manas" University in 2010.

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

Olympiads in Informatics, 2022, Vol. 16, 135–143
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.11

Informatics Olympiads in Kazakhstan: Team Selection and National Olympiads in Informatics

Yerkebulan SAGYNTAY
Doctoral School, Faculty of Informatics, Eötvös Loránd University
Budapest, Hungary
e-mail: sagyntay@inf.elte.hu

Abstract. Improving education is one of the main reasons for modern society. Therefore, subject Olympiads make it possible to identify talented schoolchildren and students to improve the quality of education. The National Olympiads of schoolchildren in informatics, which are held annually, allowing gifted students to demonstrate a high level of training in subjects and contribute to the development of mental and creative abilities. The main goals and objectives of the Olympiad in Kazakhstan are the promotion of scientific knowledge and the development of students' interest in scientific activity, the creation of the necessary conditions for identifying gifted children, their further intellectual development, the selection and preparation of students to participate in international Olympiads, support with the choice of specialty when entering universities, increasing the prestige of education in the Republic of Kazakhstan. This article describes the current model of organizing Informatics Olympiads in Kazakhstan: steps, team selection (national and international olympiads), and evaluation. This paper also includes the comparison with Informatics Olympiad organization procedure of Slovakia and Hungary.

Keywords: IOI, olympiad in informatics, teaching programming.

Introduction

Kazakhstan began active participation in the IOI in Portugal in 1998 with 4 participants. The Republican Scientific and Practical Center «Daryn» (Republican Scientific and Practical Center "Daryn") of the Ministry of Education and Science (Ministry of Education and Science of the Republic of Kazakhstan) is responsible for organizing and selecting National Computer Science Olympiads for secondary school students. The Republican Scientific and Practical Center "Daryn" (RNPC "Daryn") supports the organizational structure of the Olympiads in Kazakhstan not only in computer science, but also in other areas, such as biology, physics, chemistry, mathematics. The annual 27th International Olympiad in Informatics (IOI-2015) was held in Almaty on the basis

of Al Farabi Kazakh National University from July 26 to August 2, 2015. 324 people from 83 countries took part in the competition.

Kazakhstan is the second state among the CIS countries after Belarus, which has received the right to host the International Olympiad of Schoolchildren in Informatics.

In the following sections of this report, we will describe in detail the entire workflow of organizing for the Informatics Olympiads in Kazakhstan. We also provide statistics of medals for the last 23 years of the Kazakh IOI team.

1. Work Related

Nikhazy and Zsako (2020), in their paper, presented information about team selection, national informatics competitions, and training in Hungary. They also provided an overview of competitions in application development and usage, computational thinking and robotics.

Makieva *et al.* (2017) analyzed different problems and ways of improvement for Informatics activities in Kyrgyzstan. Researchers described The Contest Management System and suggested that it will improve the performance of students.

Forisek (2007) described olympiad preparation, team selection of Slovakian students and other important activities such as international cooperation.

In Gomel, students who prepared for the olympiad in Informatics practiced in courses such as "Programming-professionals (individual and collegiate)," "Preparing for IOI," "Methods of algorithmization," "Basic programming," "Informatics" and learned different tasks based on various programming languages (Dolinsky, 2016, p. 241).

According to Iglikov *et al.* (2013) the important factor of Kazakhstani students' success is the participation in various International competitions and camps, among them researchers emphasized the Summer school of computing for secondary school students (Summer School of Computing), Petrozavodsk training camps for high school students (Petrozavodsk Training Camps), and E. Pankratiev Open Team Programming Collegiate Cup (E. Pankratiev Open Team Programming Collegiate Cup), all these events in Russia.

Kiryukin and Tsvetkova (2011, p. 45) claim that participation in programming competitions is significant because it has many positive aspects such as persistence and the ability to sustain pressure. Olympiad preparation is an extracurricular activity, and this involvement of students relates positively to school engagement (Frederick and Eccles, 2005, p. 516). This activity is generally organized in groups and relates to the cooperative learning method, which positively influences students' social skills (Lavasani *et al.*, 2011, p.1803). Amaroli *et al.* (2018, p.133) suggest fostering Computer science education through teams olympiads.

2. Educational Olympiads in Kazakhstan

Every year, the Olympiad is held throughout the academic year in four stages by the relevant educational authorities for each general education subject among students of grades 9–11 (Fig. 1):

1) The first (school) stage is held no later than November 30 of the current academic year in secondary education organizations according to the tasks prepared by the education authorities of the district (city).
2) The second (district/city) stage is carried out by the educational authorities of the district (city) at the district or city level.
3) The third (regional) stage is carried out by the education departments of the regions, the cities of Astana, Almaty and Shymkent, republican educational organizations, the autonomous educational organization "Nazarbayev Intellectual Schools" (hereinafter – AOO NIS) and the Non-profit joint-stock Company "Physics-Mathematics State School" (hereinafter – NAO "PMSS").
4) The fourth (republican) stage is held with the division of subjects of natural-mathematical and social-humanitarian directions in different regions of the republic.

The works of the participants at all stages of the Olympiad are provided to the jury in advance in encrypted and scanned form. The evaluation of the works by the jury members is provided in accordance with the evaluation criteria developed by the organizing committees. The results of the evaluation of works at the end of each stage are transmitted by the organizing committee.

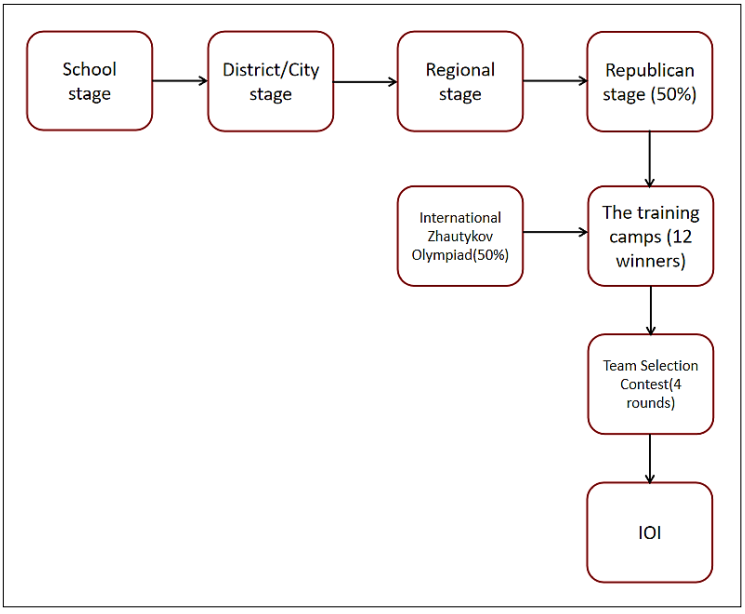


Fig. 1. Stages of Republican Olympiad in Informatics.

2.1. *International Zhautykov Olympiad*

The International Zhautyk Olympiad in Mathematics, Physics and Computer Science for students of specialized schools (hereinafter referred to as the Olympiad) is held annually in the second decade of January on the basis of the Non-profit Joint Stock Company "Physics-Mathematics State School" (hereinafter referred to as "PMSS") in Almaty, Republic of Kazakhstan. The goals and objectives of the Olympiad are: creation of optimal conditions for identifying gifted schoolchildren, their further intellectual development and professional orientation, dissemination and popularization of physical, mathematical and IT knowledge among schoolchildren, development of international cooperation with physical and mathematical schools of various countries.

Participants of the Olympiad

Teams of schools of Kazakhstan and foreign countries from specialized physics and mathematics lyceums and gymnasiums and other schools are invited to the Olympiad. Several schools from the same country can participate in the Olympiad.

Each team should consist of 7 participants: 3 participants in mathematics, 2 in physics and 2 participants in computer science and 1 or 2 supervisors (physicist, mathematician, computer scientist) who speak one of the three official languages of the Olympiad: Kazakh, Russian or English. The participation of school teams in incomplete composition is allowed.

According to the results of the Republican Olympiad of schoolchildren in the subjects of the natural and mathematical direction, twenty-one teams of the Republic of Kazakhstan are determined to participate in the Olympiad. The list of teams is approved by the organizing committee of the Olympiad in coordination with the The Republican Scientific and Practical Center «Daryn».

3. International Olympiads

International Olympiads of schoolchildren in Informatics are intellectual competitions for high school students (9th–11th grade) from different regions of the world. Each country is represented by a team of 4–6 people – winners of national Olympiads. When drawing up assignments, the diversity of world educational standards is taken into account. For a successful performance, in addition to knowledge, non-standard thinking and a creative approach to solving problems are required. Each Olympiad is a separate competition, with its own organizers, rules of conduct and an award system. International Olympiads in which Kazakhstan participates:

- International Olympiad in Informatics (IOI).
- International O. Zhautykov Olympiad in Mathematics, Physics and Informatics for students of specialized schools.
- International Junior Science Olympiad (IJSO).

- International Olympiad for schoolchildren "Tuymaada" in mathematics, physics, chemistry, informatics.
- Central European Olympiad in Informatics (CEOI).
- European Junior Olympiad in Informatics (EJOI).
- Eurasian Olympiad in Informatics (for the SCO countries).

4. Team Selection for IOI Participation

Criteria for the selection of applicants for participation in training camps for participation in the international Olympiad in the subject of informatics.

Formation of the composition of the participants in the training camps of the Olympic reserve: Based on the results of the participation of schoolchildren in the final stage of the Republican Olympiad for schoolchildren, the composition of the training camps is formed. The training camp includes 12 winners and prize-winners of the Republican Olympiad (awarded with diplomas of the first, second and third degrees) students of grades 9–10–11 (12) (Olympiad tasks are the same), who scored the most points. Applicants are selected by the sum of points for the Republican Olympiad and the International Zhautykov Olympiad (Table 1). The number can be increased if several participants scored the same number of points.

Selection to the national team for participation in the IOI: for the qualitative selection of the national team for the IOI, 4–5 intermediate olympiads will be held at the qualifying training camps (Table 2). The level of complexity of the Olympiads at the training camps is identical to the level of the IOI tasks, and the quality of the tasks is superior to the International Zhautykov Olympiad and the Republican Olympiad. Each olympiad will consist of 3 tasks.

Table 1

Formation of the composition of the participants in the training camps of the Olympic reserve

№	Full name of the participant	Grade	Results		Total
			The Republican Olympiad of schoolchildren	The International Zhautykov Olympiad	
			50%	50%	100%

Table 2

Selection to the national team for participation in the IOI

№	Full name of the participant	Grade	Results				Total
			1 round	2 round	3 round	4 round	
			25%	25%	25%	25%	100%

5. Review of Informatics Olympiad Organization Procedure in Slovakia and Hungary

Olympiad organization procedure and team selection of Kazakhstan and Slovakia is slightly similar, but different from Hungary.

Slovakia.

According to The Ministry of Education of the Slovak Republic, the Olympiad in Informatics at the national level is provided by a national commission called the Slovak Olympiad in Informatics (SK OI). The Slovak national Olympiad in Informatics (OI) has two categories: A and B. Only those students who do not graduate from high school this or the next school year may join Category B. All students (primary and secondary) can take part in category A. Category B has two rounds: domestic and regional. Category A consists of three rounds:

(1) **Home round** – organized by local teachers in schools. For each task of the home round, you can get from 0 to 10 points.
(2) **Regional round** – Slovakia has 8 regions for this round. Each region has chairmen of regional commissions. After the solution is corrected, the coordination of scoring scales will take place, the result sheets will be merged into one nationwide, and according to her, approximately the top 30 solvers are invited to the national round.
(3) **National round** – the final round which consists of two days. Participants solve theoretical problems on the first day and practical problems on the second day.

Top 10 winners will be invited to the weekly qualifying camps. And based on their results, SK OI will select teams to participate in the International Olympiad in Informatics (IOI) and the Central European Olympiad in Informatics (CEOI).

From 1993 till now, in IOI, the Slovak team has 102 medals: 25 gold, 43 silver, 34 bronze.

Hungary.

Hungary is one of the first countries to participate in the IOI since 1989. According to John von Neumann Computer Society (NJSZT), the Hungarian national team for IOI-CEOI is selected after the conduction of 6 stages. The following competitions can participate in the qualifying competition of the International Student Olympiad in Computer Science (IOI):

(1) National High School Education Competition in Informatics (OKTV) – first 15–20 places.
(2) International programming competition Noble Tihamér – first 3–7 places in 10th grade.
(3) Olympic qualifying competition (previous academic year) – 4–6 participants.
(4) Izsák Imre Gyula Competition – winner, winner in the field of information technology.
(5) Dusza Árpád Memorial Programming Contest – members of the winning team.

A member of the IOI Olympic team automatically becomes a participant who won a gold medal at the Olympic Games last year (CEOI, IOI). A member of the CEOI Olympic team automatically becomes a participant who won a gold medal at the previous year's Olympic Games (CEOI) if he or she is a student up to the 11th grade. Qualification Competition for selecting Hungarian team for the International Informatics Olympiad is intensive and consists of 6 rounds where students have to solve 18 problems in 3 days (Nikhazy and Zsako, 2020).

Stages of the qualifying competition for IOI-CEOI:

Stage 1 of the competition – with the participation of 30–40 competitors.
Stage 2 of the competition – with the participation of 30–40 competitors.
Stage 3 of the competition – with the participation of 20–25 competitors.
Stage 4 of the competition – with 15–20 competitors.
Selection of 6–6 Olympic team candidate candidates.
Stage 5 of the competition – with 6–12 competitors.
Stage 6 of the competition – with 6–12 competitors.
Selection of 4–4 Olympic team members.

Before the third day of the competition, participants will take part in online training. The official result is determined by the Competition Commission before the start of the current round.

As a result, for all the years of participation since 1989, participants from Hungary have won 13 gold, 36 silver, 46 bronze medals.

Conclusion

Kazakhstan has been participating in the IOI since 1998 and has good achievements. At the moment, Kazakh IOI teams have won a total of 61 medals (gold 3, silver 21, and bronze 37). The stages of team selection (republican and international Olympiads) are described and compared with Hungary and Slovakia. However, some issues with the Informatics Olympiads promotion exist and actions should be taken to spread essential knowledge of Informatics by involving more students and schools.

Table 3
Performance in IOI

Year	IOI Host	Medals			
		Gold	Silver	Bronze	Total
1998	Portugal				
1999	Turkey			1	1
2000	China				
2001	Finland			2	2
2002	South Korea				

Continued on next page

Table 3 – continued from previous page

Year	IOI Host	Medals			
		Gold	Silver	Bronze	Total
2003	USA			1	1
2004	Greece		1	2	3
2005	Poland			3	3
2006	Mexico		1	3	4
2007	Croatia	2	1	1	4
2008	Egypt		2		2
2009	Bulgaria			2	2
2010	Canada			3	3
2011	Thailand		1	3	4
2012	Italy		3		3
2013	Australia			1	1
2014	Taiwan		2	1	3
2015	Kazakhstan		3	1	4
2016	Russia		3	1	4
2017	Iran		1	3	4
2018	Japan		1	3	4
2019	Azerbaijan	1	1	2	4
2020	Singapore		1		1
2021	Singapore			4	4
Total		**3**	**21**	**37**	**61**

References

Amaroli, N., Audrito, G., Laura, L. (2018). Fostering informatics education through teams olympiad. In: *30th International Olympiad in Informatics, IOI 2018* (Vol. 12, pp. 133–146).

Dolinsky, M. (2016). Gomel training school for Olympiads in Informatics. *Olympiads in Informatics*, 10, 237–247.

Iglikov, A., Gamezardashvili, Z., Matkarimov, B. (2013). International olympiads in informatics in Kazakhstan. *Olympiads in Informatics*, 7, 153–162.

Forišek, M. (2007). Slovak IOI 2007 team selection and preparation. *Olympiads in Informatics*, 1, 57–65.

Kiryukhin, V., Tsvetkova, M. (2011). Preparing for the IOI through Developmental Teaching. *Olympiads in Informatics*, 5, 44–57.

Lavasani, M. G., Afzali, L., Afzali, F. (2011). Cooperative learning and social skills. Cypriot *Journal of Educational Sciences*, 6(4), 186–193.

Makieva, Z., Khalikov, F., Alimbaev, R. (2017). Kyrgyzstan Olympiad in Informatics: Training Students, Conducting the Olympiad and Using Contest Management System. *Olympiads in Informatics*, l, 159–166.

Nikházy, L., Zsakó, L. (2020). National Programming Competitions, Team Selection and Training in Hungary.

Order of the Minister of Education and Science of the Republic of Kazakhstan dated December 7, 2011 No. 514. Registered with the Ministry of Justice of the Republic of Kazakhstan on December 27, 2011 No. 7355.

Criteria for the selection of applicants for participation in training camps for participation in the international Olympiad in the subject of informatics. `https://daryn.kz/docs/информатика.pdf`

IOI. International Olympiad in Informatics. `https://ioinformatics.org/`

The Republican Scientific and Practical Center «Daryn». `https://daryn.kz/`

John von Neumann Computer Society (NJSZT). `https://njszt.hu/`

Olympiáda v informatike. `http://oi.sk/`

Y. Sagyntay received his MSc degree in computer science engineering from Obuda University, Hungary, and is currently studying his Ph.D. at the computer science at the Eötvös Loránd University in Hungary. His main research interests are education in computer science and algorithms.

Olympiads in Informatics, 2022, Vol. 16, 145–158
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.12

Organization and Results of Mongolian National Online Olympiads in Informatics

Danzan TSEDEVSUREN[1], Jantsansambuu DASHDEMBEREL[1], Tsiyen-Oidov BATTOGTOKH[1], Turtogtokh ULAMBAYAR[1], Altangerel KHUDER[2]

[1]*Mongolian State University of Education, School of Mathematics and Natural Science, Department of Informatics*
[2]*Mongolian University of Science and Technology, School of Information and Communication Technology, Department of Computer Science*
e-mail: tsedevsuren@msue.edu.mn, dashdemberel@msue.edu.mn, battogtokh@msue.edu.mn, ulambayar@msue.edu.mn, khuder@must.edu.mn

Abstract. Incorporating coding skills into the basic literacy skills of 21st century citizens is common in many parts of the world. This is because the development of artificial intelligence and smart devices and their social use have become real, and in the near future, the ability to use robots and artificial intelligence devices for their own purposes has become a skill that every citizen should have. Algorithms and programming are included in the Mongolian general education information technology course curriculum. The coding ability plays an important role in the development of a new century citizen's thinking, creating and evaluating skills. One of the activities that promotes the development of this skill is the International Olympiad in Informatics. Our country has been participating in this Olympiad since 1991 and has won three bronze, one silver and one gold medal. You can participate in the online Olympiad regardless of where you live in Mongolia. This type of Olympiad is very important to support the continuous development of students who are gifted in programming and coding, as well as to enable them to successfully participate in national and international Olympiads.

Mongolian Informatics Olympiad Committee (MIOC) organized 24 online contests using Contest Management System (CMS[1]) which is official IOI judging system. In this paper we considered 22 online contests organized in 2019, 2020, 2021 years and classified 115 problems chosen in those contests by topics and complexity. We also report here results of a small research about scores got by participants, development of problem-solving skills. A new registration web system developed while implementing IOI judging system is explained.

Keywords: informatics, programming, grading system, olympiad, online judge system.

[1] Open-source contest management system. `https://cms-dev.github.io/`

Introduction

Contest Management System (CMS) is a system developed by Italian software engineers and there was a successful localization for Mongolian language in 2015. Now it is being used in following olympiads.

- Among high school teachers and students:
  - District level olympiads for 9 districts in Ulaanbaatar city.
  - 21 provinces and capital city informatics olympiads.
  - National Olympiads in Informatics.
  - Contests for selecting IOI participants.
  - Online olympiads.
- Among university students:
  - Algor-ithm and programming related courses.
  - Programming olympiads inside a university.
  - State level Programming Olympiad for University students.

In order for the competition to be successful, it is important to support participation of people by connecting people with similar interests such as informatics and problem-

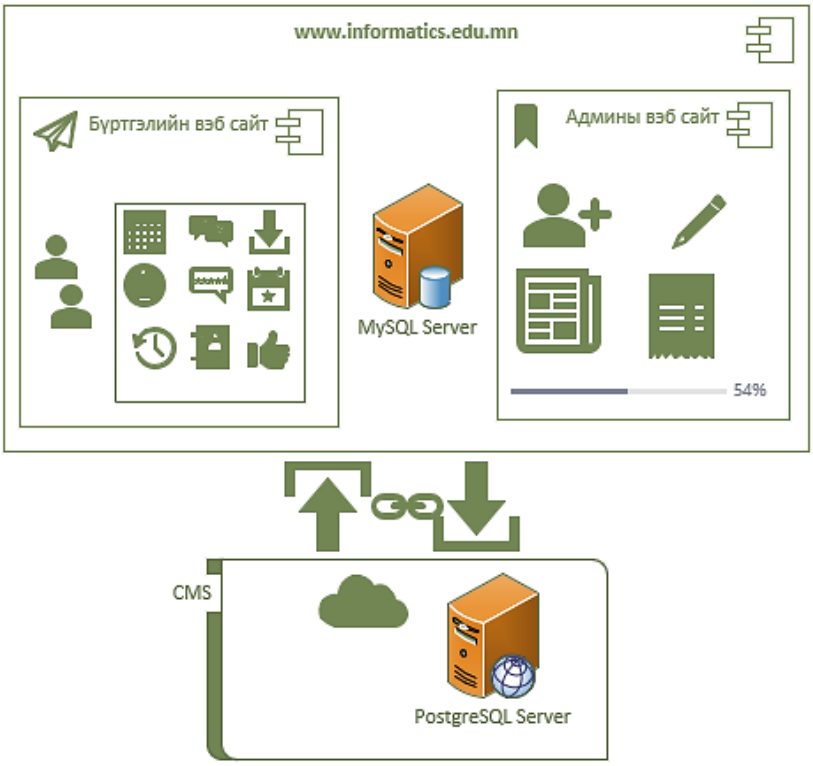


Fig. 1. Cooperation of CMS and Informatics olympiad registration website.

solving. We must pay attention to provide a good networking environment among people interested in programming from different areas of study therefore to improve their chances to develop together. (Amaroli, Audrito & Laura, 2018).

Development of a web registration system used for olympiad participants, alumni's, statistical processing, exporting information for CMS started in December 2017 and finished in January 2018.

The website `www.informatics.edu.mn/burtgel` consists of participant registration page and admin page. It is currently being developed continuously (Dashdemberel & Ulambayar, 2017).

These systems were designed to work with CMS and official MIOC website. Our next step is developing a continuous online contest system and it is being tested in local environment. These two websites have shared database and both exchange information with CMS while working.

These are the two systems we have developed.

About CMS v1.4

Since 2015 we are using CMS (`http://cms-dev.github.io/`) (Maggiolo and Mascellani, 2012; Maggiolo *et al.*, 2014) for each level of programming and informatics olympiads in Mongolia.

We use CMS 1.4 for online and offline contests. Fig. 2 shows basic CMS operations (Maggiolo and Mascellani, 2012).

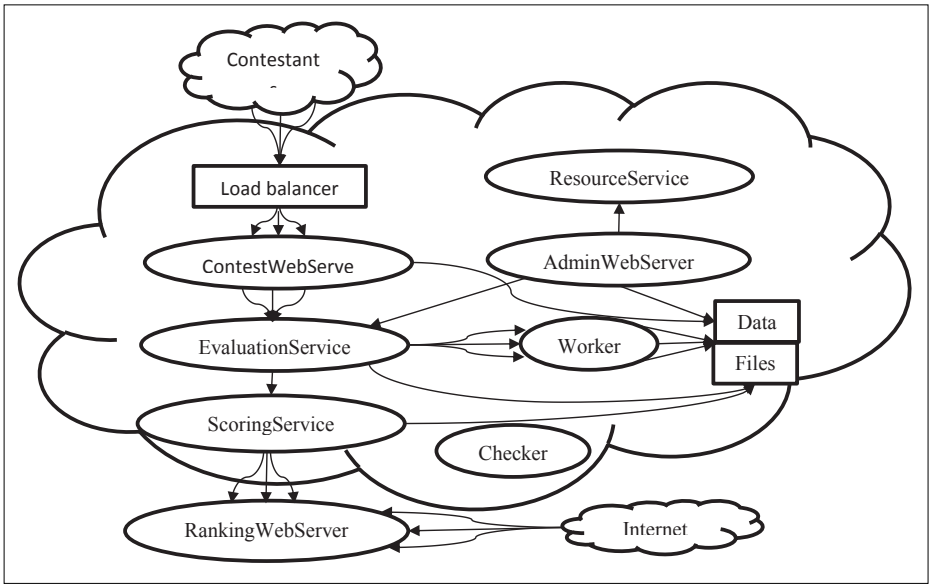


Fig. 2. Basic CMS operations.
(Source: Maggiolo and Mascellani, 2012)

AdminWebServer is a admin web service to provide contest organizers with operations such as insert, edit participants' information, view participants' solutions, contest ranking and download them.

ContestWebServer has mainly participant operations such as get contest information, download problem statements, ask questions about a problem, send solutions, view own scores. This server program is duplicated and loaded on several servers using load balancer when the contest size is big.

EvaluationService distributes contestants' solutions to Worker threads to check them. Then it takes results of each test and sends it to ***ScoringService*** service. Each Worker thread gets participant code, recognizes programming language used, compiles the code using corresponding compiler to get an executable, runs it to get results of each test and writes it to the database.

ScoringService combines evaluated scores for each test from ***EvaluationService*** and sends total score to ***RankingWebServer*** web page. ***RankingWebServer*** lists all participant scores and publishes the list on website.

There were following additional requirements in process of localization.

1. Automatically register, create passwords, insert into CMS system, create certificate for each contest. This leaded us to develop our website.
2. Create problem archive after classifying problems by type and level. Register users on the web, change rank list view, organize open contests.
3. Create beginner, middle, advanced level training website to increase participant count. Students will be able to send request for training material and improve their skills. This kind of training website can be replaced by Moodle LMS.

Focusing above requirements we have developed a registration website which cooperates with CMS system.

One can read about organization of programming and informatics contests among high school students, improvement of students' participation in this kind of contests, online learning platforms for computer science courses in many papers by international researchers (William, Gabriele, Luigi, Umberto, Marco & Luca, 2016).

Our web system consists of administration and user registration sections.

a. Administrator website v2.0.

Actions allowed for admin user

- Manage registered users (Add into active contest as a contestant, enable or disable log in permission, remove participant).
- Check registered participants' information of an announced contest and confirm or cancel contest participation requests.
- Automatically create username and password for CMS system and email them to confirmed participants. Publish usernames on the website.
- Send e-mail to users.

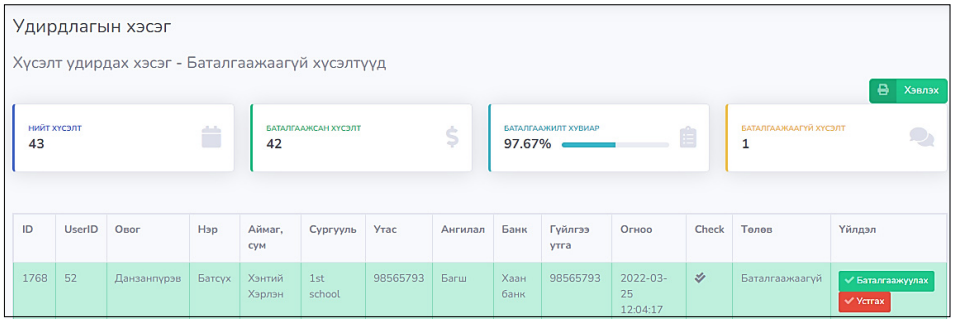


Fig. 3. Control panel of admin website.

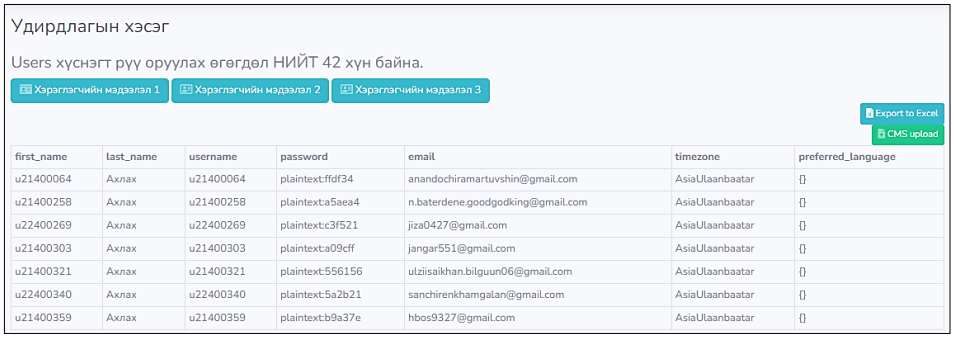


Fig. 4. User list.

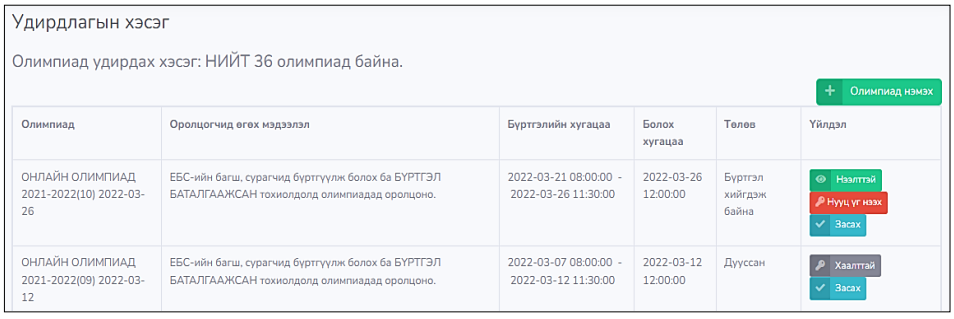


Fig. 5. Contest list.

- Prepare data for a CMS contest and export. Upload teachers' and students' information into CMS.
- Manage contests (add, activate, open, close).
- Add, manage additional materials for contestants.

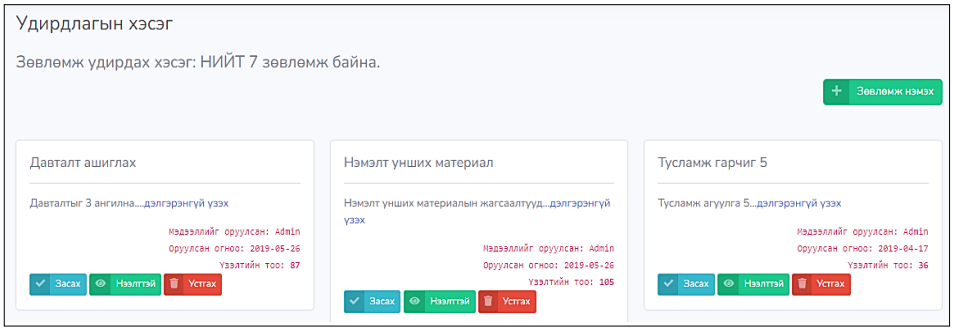


Fig. 6. Additional materials page.

b. Registration website v2.0

Actions allowed for users

- Register, log in, restore password, change password.
- Send request to active contest. In case of confirmation get username and password.
- View contest rank list, problem statements, solutions and problem statistics. Download tests for a problem.
- Download problem statements, solutions and tests from problem set.
- View additional materials.
- Go to additional olympiad problem sets.
- Download certificate of participation.

The registration website also has some extra pages for regulations, training materials, problem set, certificates. The problem set consists of 450 problems. Users can download problem statements, solutions, tests. Problems are classified into 4 complexity groups

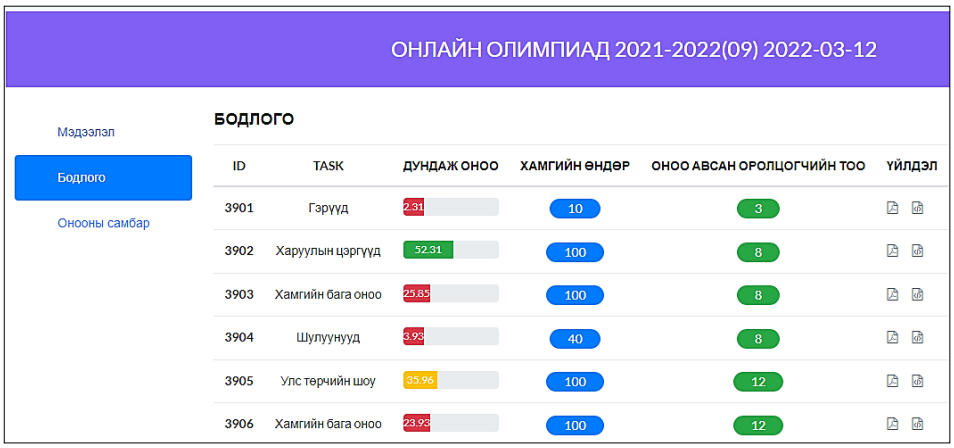


Fig. 7. Problems list.

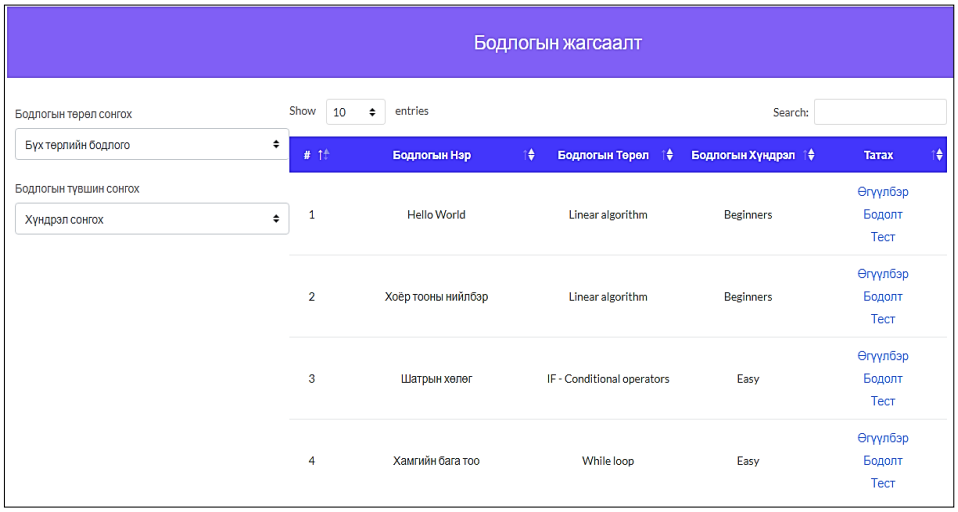


Fig. 8. Problem statement, solution and tests list.

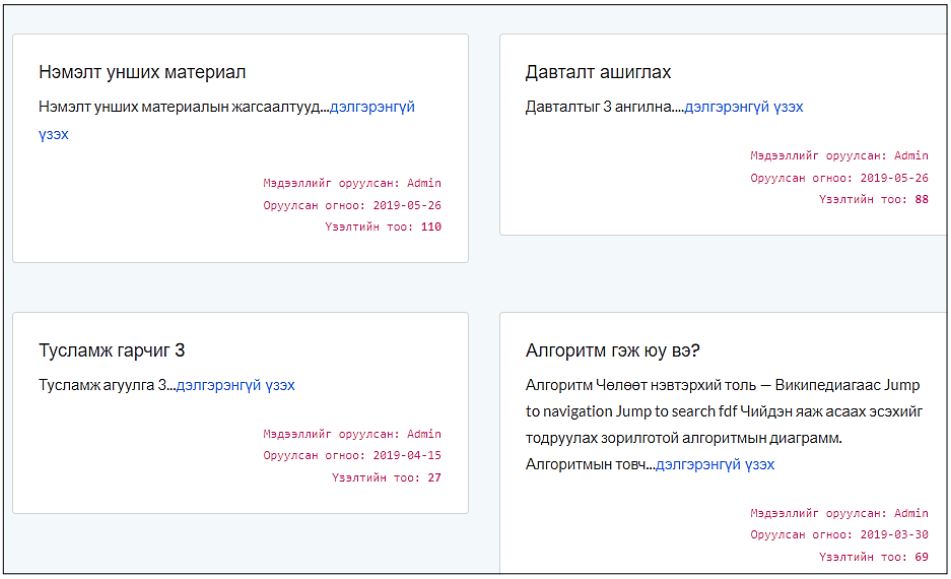


Fig. 9. Additional materials page.

and 25 topics. These problems can be used by beginners, olympiad participants. Also, they will be useful in programming and algorithm courses of Information technology, Computer Science, Software Engineering undergraduate programs.

Influences of the Online Olympiads

Highschool students are facing several difficulties due to their English language barrier. The most widely spread difficulties are being not able to participate in online programming contests in English, using online resources in English, difficulties with understanding problems in English etc. Regular participation in online contests helps them make programming and algorithmic skills better. Also, online contest rankings of our students show us their readiness for international level competitions (Khuder & Tsedevsuren, 2016).

The top informatics olympiad skills are algorithmic skills, self-study, using programming tools, digital8 technological and technical skills, communication skills and creativity (Tsvetkova, Kiryukhin, 2020). It is very important to organize regular online contests which are considered as exercise environment for developing these skills.

The most important information source about above topics is the proceeding of IOI conference – "Olympiads in Informatics" (international forum for presenting research and development in the specific area of teaching and learning informatics through competition) first publication of which was in 2007. Books such as (Skiena and Revilla, 2003) and (Halim and Halim, 2013) includes important materials about programming contests, algorithms, data structures and computer science (William, Gabriele, Luigi, Umberto, Marco & Luca, 2016).

Dagienė (Dagienė, 2010), Garcia-Mateos and Fernandez-Aleman (Garcia-Mateos and Fernandez-Aleman, 2009) noted about importance and influence of programming, computer science olympiads in studying computer science.

The core element and skill of programming education is basic coding skills which includes programming according to programming language syntax and problem solving. Students should learn both basic algorithms and their implementations. There are two basic types of errors in code: syntax and static sematic errors, dynamic semantic errors. While errors of first type are discovered by compiler, for the second type errors require testing. Students should improve their skills of making tests.

First online open contest was organized in 15th of March, 2018 and then we tested these webpages. Here we showed only main statistics. Each user registers with his email and email defines unique user. We send confirmation email and after user confirms it he or she will be able to use the system. Now we have 721 users in our contest registration website (51 of them did not confirm their email). Hence, we have around 670 active users.

There were 495 participants from Ulaanbaatar city, 175 participants from provinces. Top 4 provinces by participant count were Uvs (43), Uvurkhangai (26), Darkhan city (20) and Bayankhongor (11). Average participant count among 20 provinces was 8,75. Recent years' top provinces by participant scores in National Olympiads in Informatics are Uvs, Darkhan city, Bayankhongor, Khubsugul, and Khobdo. Participants visiting statistic was between 1 and 192. Average visit count was 11,4. Since the website was created there were made 7692 visits. There were 735, 1378, 1168, 4411 visits in years 2018, 2019, 2020 and 2021. Visiting count of the webpage for the first week of January, 2022 is 532.

Table 1
Classification of contestants

Classification	Count
Teacher	79
Senior	132
Secondary	78
Other	27
Total	**316**

Table 2
Participants and problem statistics

Year	Olympiads organized	Contestant count					Problems	
		Teacher	Senior	Secondary	Other	Total	Teacher	Student
2019	4	46	50	25	25	**146**	4	4
2020	8	8	13	1	7	**29**	5	5
2021	10	13	9	10		**32**	4	4
Total	22	317	386	172	29	**904**	82	84

As of today, we have organized 24 online contests in total. There were 2 contests in 2018, 4 in 2019, 8 in 2020 10 in 2021. This paper covers results and analysis of 22 online contests from 2019, 2020, 2021 years. A total of 904 teachers and students participated in the 22 Olympiads, 115 problems were proposed and a database of results was formed.

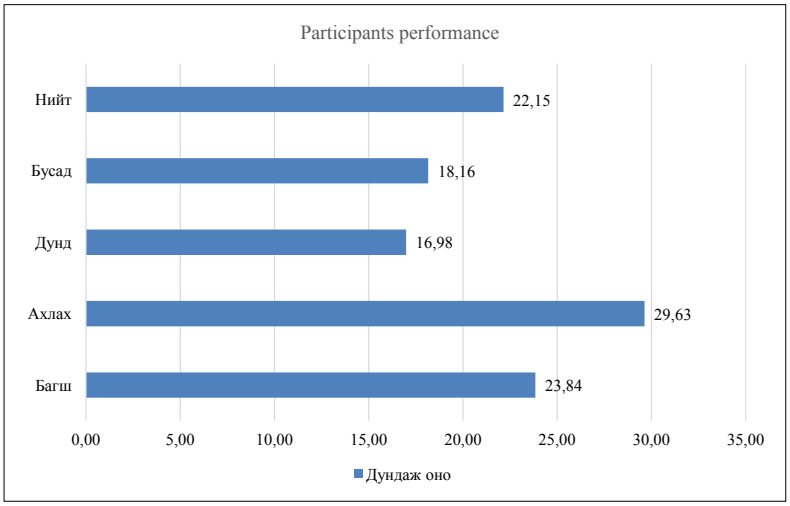
Statistical information shows there are 79 teachers and 210 students among participants. Recent years amount of teachers increased and it also make amount of students interested in programming. Reason of this may be scholarships in foreign universities and former olympiad participants who works now in world level IT companies such as Google, Facebook, Amazon, Microsoft. Total amount of participants in 22 online contests was 316. Table 1 shows number of participants by classification.

Participant count of our online olympiad was between 11 and 77. There were 22 online olympiads organized and in average there were 18 teachers, 22 students in each contest (average contestant count was 41). Each contest has 3–4 problems. There were 10 online contests in 2021 and each has two categories: teachers and students. After adding category "Teachers" number of teacher-participants is steadily increasing.

Above performance statistics show us the Senior students get the best scores. Also, we can see the average performance of teachers and senior students are higher than the general average by 1.7 and 7.5 percent correspondingly.

115 problems used in contests were classified into 4 levels and there were 4 easy level problems, 30 middle level, 51 hard level and 30 advanced level problems. We can see the average score of performance was decreasing with the increasing level of problem.

Performance per complexity of the problems is shown in Table 3.



Graphic 1. Participants performance.

Table 3
Performance by problem complexity

Problem complexity	Full solution	50–99 scores	Less than 50 scores	0 scores	Average score
Easy	38	38	13	29	60.28
Middle	191	118	147	435	34.97
Hard	208	98	248	807	23.85
Advanced	56	38	110	654	11.55
	493	292	518	1925	32.6625

Table 4
Time and performance

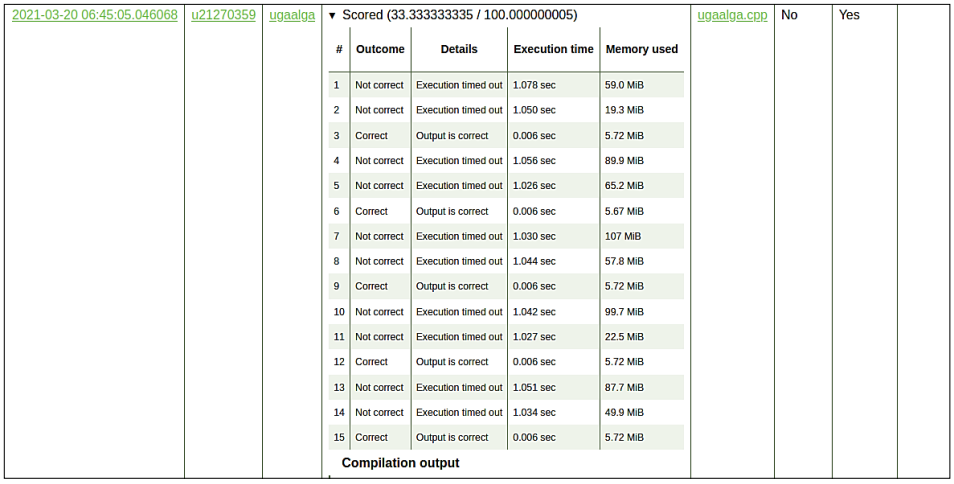
Time (c)	Problem count	Full solutions	50–99 scores	Less than 50 scores	0 scores	Average score
0.1	3	10	2	16	63	14.76
0.5	41	125	57	134	393	26.99
1	55	258	116	221	850	26.95
1.5	2	1	12	2	51	11.29
2	10	10	5	23	136	10.22
2.5	1	0	2	2	23	7.96
3	3	14	28	19	41	36.17
Total	**115**	**493**	**292**	**518**	**1925**	**19.85**

Time and memory limits are main settings for a informatics problem. We used mostly time limits for problems and memory limits are not widely used. Running times of participant solutions are shown in Table 4.

We can see from the above table that from all solutions there are full solutions – 16.0%, solutions which got between 50–99 points – 8.5%, solutions with less than 50 points – 16.0%.

All 115 problems used in online contests can be classified into 12 classes. Table 5 shows performance in 4 levels for each class of problems.

Our problem classification matches with important topics in IOI syllabus. We should develop training materials according to IOI syllabus. Insufficient knowledge and skills from IOI syllabus leads to poor planned training and unsuccessful IOI participation (Khuder, Tsedevsuren, 2016). Therefore we should pay attention to improve those skills of students which gives us bad average score.



2021-03-20 06:45:05.046068 | u21270359 | ugaalga | ▾ Scored (33.333333335 / 100.000000005) | ugaalga.cpp | No | Yes

#	Outcome	Details	Execution time	Memory used
1	Not correct	Execution timed out	1.078 sec	59.0 MiB
2	Not correct	Execution timed out	1.050 sec	19.3 MiB
3	Correct	Output is correct	0.006 sec	5.72 MiB
4	Not correct	Execution timed out	1.056 sec	89.9 MiB
5	Not correct	Execution timed out	1.026 sec	65.2 MiB
6	Correct	Output is correct	0.006 sec	5.67 MiB
7	Not correct	Execution timed out	1.030 sec	107 MiB
8	Not correct	Execution timed out	1.044 sec	57.8 MiB
9	Correct	Output is correct	0.006 sec	5.72 MiB
10	Not correct	Execution timed out	1.042 sec	99.7 MiB
11	Not correct	Execution timed out	1.027 sec	22.5 MiB
12	Correct	Output is correct	0.006 sec	5.72 MiB
13	Not correct	Execution timed out	1.051 sec	87.7 MiB
14	Not correct	Execution timed out	1.034 sec	49.9 MiB
15	Correct	Output is correct	0.006 sec	5.72 MiB

Compilation output

Fig. 10. CMS view of a participant solution.

Table 5

Problem classification and performance

Classification	Full solutions	50–99 scores	Less than 50 scores	0 scores	Average score	Problem count
Linear algorithm	14	1	0	5	72.5	1
Number theory	158	125	113	282	39.79	27
Strings	27	16	57	94	28.39	8
Dynamic programming	47	36	106	253	21.27	14
Geometry	85	40	97	413	20.97	24
Sequence and sorting	80	41	57	409	20.39	20
Other	15	2	8	61	20.02	3
BFS	12	2	21	55	17.00	2
2D array	10	3	13	44	18.93	4
DFS	18	7	19	112	17.74	5
Graph theory	27	19	27	197	17.07	7
Total	493	292	518	1925	25.92	115

Батламжууд

#	ОЛИМПИАД	АВСАН ОНОО	ҮЙЛДЭЛ
1	ОНЛАЙН ОЛИМПИАД 2020-07 2020-04-17	260	ТАТАХ
2	ОНЛАЙН ОЛИМПИАД 2020-06 2020-04-10	310	ТАТАХ
3	ОНЛАЙН ОЛИМПИАД 2020-05 2020-04-03	300	ТАТАХ
4	ОНЛАЙН ОЛИМПИАД 2020-04 2020-03-27	400	ТАТАХ
5	ОНЛАЙН ОЛИМПИАД 2020-03 2020-03-21	378	ТАТАХ

Fig. 11. Certificate list.

Conclusion

After regular online olympiads participants' problem solving and technical skills are improving. There were no "Teacher" category in 2018–2020 and not many teachers participated but after adding the category teacher count has increased. We can also set up categories "beginner", "middle" and "advanced" in registration and CMS webpage.

Organizing regular online olympiads increases interest in "Competitive programming, participant number and also student numbers studying algorithms. Now we are going to define hard topics for students and create online learning content about them. This will help us improve general programming skill level of participants. There are additional online learning materials for dynamic programming, graph algorithms, computational geometry created by teachers and published for students. Another important result of regular online contests is practicing and improvement in time management, learning to choose which problem to try first in IOI.

Mongolian IOI team got IOI medals for past 4 years. We conclude that our online contests have some influence in those successful participations. Specifically, Tenuun got bronze medal in 2018, Nyamdavaa got two silver medals in 2019 and 2021. He also got his gold medal in 2020. In total Mongolian team got 6 medals from IOI.

We strongly believe that organizing regular online contests can be strong educational support for improving coding and algorithmic thinking skills for informatics teachers and students.

References

Amaroli, N, Audrito, G, Laura, L. (2018). Fostering informatics education through teams Olympiad. *Olympiads in Informatics*, 12, 133–146.

Dagienė, V. (2010). Sustaining informatics education by contests. In: *Teaching Fundamentals Concepts of Informatics*. Springer, 1–12.

Dashdemberel, J., Ulambayar, T. (2017). Using assessment CMS of informatics problems. *Journal of Mathematics & Natural Science*, MNUE, Vol. 3.
Garcia-Mateos, G., Fernandez-Aleman, J.L. (2009). Make learning fun with programming contests. In: *Transactions on Edutainment II.* Springer, 246–257.
Halim, S., Halim, F. (2013). *Competitive Programming.* Third Edition. Lulu.com.
Khuder A., Tsedevsuren, D. (2016). The Informatics Olympiad in Mongolia: Training resources for non-English Speaking Students. *Olympiad in Informatics*, 10, 279–284.
Maggiolo, S., Mascellani, G., Wehrstedt, L. (2014). Cms: a growing grading system. *Olympiads in Informatics*, Vol 8, pp: 123–131.
Maggiolo, S., Mascellani, G. (2012). Introducing cms: a contest management system. *Olympiads in Informatics*, 6, 86–99.
Skiena, S.S., Revilla, M.A. (2003). *Programming Challenges: The Programming Contest Training Manual*. Springer Science & Business Media.
Tsvetkova, S., Kiryukhin, M. (2020). Top 10 key skills in Olympiad in Informatics. *Olympiads in Informatics*, 14, 151–167.
William, D., Gabriele, F., Luigi, L., Umberto, N., Marco, T., Luca, V. (2016). oii-web: An interactive online programming contest training system. *Olympiads in Informatics*, 10, 207–222.

Online resources

Dashboard with System's metrics for Mongolian Online Olympiad. URL:
`https://informatics.edu.mn/burtgel`
Dashboard with System's metrics for Russian Programming Olympiad. URL:
`https://olympiads.ru`
Contest of codefoeces.com online judge. URL:
`https://codeforces.com/contests`
Contest of e-olymp online judge. URL:
`https://www.e-olymp.com/en/contests`
Achievements in Mongolia's IOI. URL:
`http://stats.ioinformatics.org/results/MNG`

D. Tsedevsuren is Professor at School of Mathematics and Natural Sciences, Mongolian National University of Education. He is PhD in ICT and Educational Studies, and he is currently working as a President of this Mongolian Informatics Association. His research interests include Informatics education, ICT in eduaction, theory and methodology of eLarning and electronic learning content development.

J. Dashdemberel is Lecturer at School of Mathematics and Natural Sciences, Mongolian National University of Education. He is M.S in ICT and Educational Studies, and he is currently working as a Member of Mongolian Informatics Association. His research interests include Informatics education, ICT in eduaction, theory and methodology of algorithm and programming learning online content development.

T. Battogtokh is Head of Informatics Department at School of Mathematics and Natural Sciences, Mongolian National University of Education. He is M.S in ICT and Educational Studies, and he is currently working as a Member of Mongolian Informatics Association. His research interests include theory and methodology of Network technology and database.

T. Ulambayar is Lecturer at School of Mathematics and Natural Sciences, Mongolian National University of Education. He is M.S in ICT and Educational Studies, and he is currently working as a Member of Mongolian Informatics Association. His research interests include and computer graphic and web design.

A. Khuder is involved in the training of the Mongolian team for the IOI since 2006, and since 2008 is the deputy team leader of the Mongolian team. He got a master degree in ITMO University and a PhD degree in Computer Science at Mongolian University of Science and Technology. He is Head of Computer Science Department in Mongolian University of Science and Technology.

Olympiads in Informatics, 2022, Vol. 16, 159–172
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.13

Digital Literacy in Primary School

Marina S. TSVETKOVA[1], Elena A. BONDARENKO[2],
Irina Yu. KHLOBYSTOVA[3], Ekaterina V. YAKUSHINA[4]

[1]*Russian Academy of Natural History, Russian Federation, Moscow, 105037, box 47*
[2]*All-Russian State Institute of Cinematography named after S. Gerasimov*
Russian Federation, Moscow
[3]*Glazovsky state pedagogical Institute named after V.G. Korolenko*
Russian Federation, Glazov, 427620
[4]*Moscow City University, Russian Federation*
Moscow, 129226, 4 Vtoroy Selskohoziajstvenny proezd
e-mail: ms-tsv@mail.ru, letty3@yandex.ru, hloirina@mail.ru, yakushinaev@mgpu.ru

Abstract. The program reveals the goals of mastering primary digital literacy by students in primary school, characterizes digital skills and determines the place of the digital literacy course in the curriculum, reveals the main approaches to the selection of course content and thematic planning, taking into account the hours chosen by the educational organization.

Keywords: educational program, primary general education, digital literacy, digital transformation of education.

1. Introduction

The sample curriculum for the subject “Digital Literacy” was developed on the basis of the Federal State Standard of Primary General Education (MERF, 2021, 2022), as well as taking into account the Federal Law on Education (FLERF , 2012) and the Strategic Direction of the Transformation of General Education in the Russian Federation (GRF, 2021).

The program sets the planned results of mastering the initial digital literacy. The program determines the content of the training course by years of study, indicating approximate hours on each topic for grades 3–4 and types of educational activities using tools of information and communication technologies.

2. Objectives of the Course "Digital Literacy" in Primary School

The program of the training course "Digital Literacy" reflects the formation of children's ideas about the high level of development of modern information technologies and their implementation in everyday life, the formation of the initial culture of the user of modern information and communication technologies, the expansion of opportunities for individual development of children through the implementation of individual curricula in an electronic learning environment using digital skills.

The content of the course reflects the skills of working with information and on the formation of information and communication technology competence as the initial digital literacy of children.

The program is focused on supporting the interdisciplinary work of students with the means of information and communication technologies in the development of programs of all academic subjects. As part of the formation of digital information skills for all subjects, it is important to develop the skills of entering, analyzing, processing, presenting, searching, transmitting digital information by computer means, and the practical use of computers, digital educational equipment, electronic information and educational environment resources and e-learning, applied means of information and communication technologies by students in the conditions of digital transformation education and the emergence of new means of information and communication technologies, that is, the need to prepare primary school graduates for competent information activities in their studies, creativity and life.

In accordance with this purposes, the course «Digital Literacy» is aimed at achieving the following goals:

- The use of digital tools and resources, tools for solving a variety of cognitive and practical tasks covering the content of all subjects studied – information retrieval; fixation (recording) information using various technical means; structuring of information, its organization and presentation in the form of simple presentations, digital information objects.
- Development of initial digital computer skills, the use of various digital learning devices, the inclusion of a computer to work with digital information of various types.
- Use of e-learning resources and digital educational environment for digital adaptation of primary school children in compliance with sanitary and hygienic standards and requirements when working with electronic learning tools.
- Formation of communicative universal educational actions: exchange of media messages; performance with audiovisual support; recording the progress of collective/personal communication; communication in a digital environment (e-mail, chat, videoconference, forum, blog, personal account, electronic diary, online contests and Olympiads) with the participation of a teacher and compliance with information security standards in a digital environment.
- Using the results of children's work posted in the information environment to evaluate and correct the work done digitally; creating a digital portfolio of personal educational achievements of the child.

2.1. *The Place of the Course "Initial Digital Literacy" in the Curriculum*

The course program is designed for a study load of 70 hours in grades 3 and 4 (based on the experience of mastering the primary skills of writing and reading texts, working with mathematical information in grades 1–2 by students).

The training course supports the practical work of children with information and communication technologies in all subjects of primary general education, including in interdisciplinary project work based on information and communication technologies.

2.2. *General Characteristics of the Course "Digital Literacy"*

Primary digital literacy is the ability to solve educational tasks using tools that are generally available in primary school, information and communication technologies on a computer, the competent inclusion of digital information and digital communication tools in the context of educational and cognitive activity in accordance with the age needs and capabilities of a younger student.

The training course should not infringe on the rights of those students who study the course in educational organizations with different levels of equipment of the information educational environment, therefore, the training course program is implemented by an educational organization taking into account the level of equipment of the information and educational environment of the educational organization and based on standard equipment, such as a student's computer and/or a teacher's computer workplace with presentation equipment, a printer and a webcam, as well as other devices connected to a computer, including digital laboratories for children.

Additionally, it is possible to use personal digital devices for photo, audio and video recording and data transmission.

3. The Main Digital Competencies of Primary School Students

The structure of digital literacy of primary school students includes three groups of digital competencies:

1. **Information literacy.** (Section of the course Working with digital information): basic skills of working with text, graphics and presentations, design, input, output, fixing and editing information on a computer and using publicly available digital devices, working with hypertext, design of creative works using a computer, presentation, algorithms and control of the screen performer.
2. **Computer literacy.** (Section of the course Computer Practice and digital technology): a set of simple skills to put into practice a computer, conventional digital devices and control interfaces for these devices (as well as for children with disabilities – special digital devices and software services), the ability to turn on the computer and select a program, recognize the menu in the program and pro-

gram tools, apply experience with devices connected to a computer, the ability to use the on-screen user interface, keyboard interface, keyboard input, remote control interface, voice interface, touch interface for special devices, work with external media for storing information (in files folders), hygienic noms of the organization of work with a digital device (preventive physical exercises, culture of norms of time, breaks, workplace organization).

3. **Communication Literacy.** (Digital Communications course section): the ability to work in computer networks to search and exchange information, to work in online educational services, with e-mail, personal account and electronic diary, class forum, school website, video services for educational activities, electronic educational resources on the computer and on the Internet, to understand the purpose of registration, password and login.

Safe behavior in the digital environment is presented in primary school by a separate training course "Information Security" as the most important component of digital culture.

Thus, digital culture includes digital literacy of children (Fgosteestr, 2022) for teaching children modern digital skills and media culture (Information Security course at school) for the socialization of children in the new digital world (Fgosteestr, 2022).

3.1. *Planned Personal and meta- Subject Competencies for the Course "Digital Literacy"*

Personal competencies (initial life experience of digital activity):

- To be aware of the need to master digital literacy in order to adapt to life situations, to develop a common human information culture; to develop information activities.
- Apply the rules of collective information activity with peers, show the ability to negotiate, lead, follow instructions, be aware of personal responsibility and objectively assess their contribution to the overall result.
- To master the skills of information activity on the basis of safe behavior in the information environment, to observe the norms of culture, respect for work and collective activity.
- Apply digital competencies to solve practical problems in everyday life, including when helping classmates, young children, adults and the elderly.
- Work in situations that expand the experience of applying digital skills in real life, increase interest in learning new things and creativity by means of modern information and communication technologies.
- Evaluate practical and educational situations from the point of view of the possibility of applying digital literacy competencies for rational and effective solutions to educational and life problems.
- Evaluate their success in information activities, outline ways to eliminate difficulties; strive to deepen their digital competencies, taking into account the legal norms of behavior.

- It is reasonable and safe to use a variety of information tools to solve the proposed and independently selected educational problems and tasks.

Meta-subject competencies (initial practical experience of digital educational work as a set of cognitive, communicative and regulatory universal educational actions) – the ability of students to use in practice the digital skills that make up:

- Universal educational cognitive actions – substitution, modeling, encoding and decoding of information, logical operations, including general methods of solving problems by means of information and communication technologies.
- Universal educational communicative actions – to organize and carry out cooperation by means of digital communications, to adequately transmit information and display the subject content and conditions of activity and speech, to argue and justify their position, to ask questions necessary for the organization of their own activities and cooperation with a partner using means of communication.
- Universal regulatory actions – in information activities, to accept and preserve the educational goal and task, plan its implementation, monitor and evaluate their actions, make appropriate adjustments to their implementation, carry out ascertaining and anticipating control over the result and method of action, actual control at the level of arbitrary attention.

These meta-subject results determine the elements of the educational experience of problem solving and creative activity of different directions.

3.2. *Key Functional Skills of Students: Children's Digital Skills in Three Types of Competencies*

Information literacy. The section "Working with digital information" will teach the student:

- To distinguish between different types of information (text, numeric, graphic, audio), to understand the peculiarity of digital information.
- To navigate in the ways of presenting and storing information on a computer and digital environment.
- Use methods of searching for information on the table of contents in the electronic catalog of the digital resource and in the Internet search engine.
- Understand and fill in simple ready-made tables; read simple ready-made bar charts in electronic text.
- Use additional sources of information on electronic media, including the controlled Internet, to find the necessary information, analyze it.
- Work with digital information: type small texts (letters, greetings, and other small texts for specific communication situations), create graphic and small multimedia objects (audio, video, presentation slides) to present, design and transmit information.
- Enter information into a computer using various technical means (keyboard, graphics tools, photo and video cameras, microphone, etc.).

- Save the information received.
- Type small texts in your native / foreign language in the simplest text editor.
- Use the basic functions of a standard text editor, follow the basic rules of text formatting.
- Draw (create simple images) in a graphic editor.
- Create simple images using the graphical capabilities of a computer; design an image from ready-made fragments (application, collage).
- Edit texts, images, slides, video and audio recordings, photo images in accordance with a communicative or educational task.
- Prepare and conduct a presentation in front of a small audience: create a presentation plan, choose audio-visual support, write explanations and abstracts for the presentation, use simple diagrams, diagrams, plans, etc.

Computer literacy. Section "Digital technology and computer practice". The student will learn:

- To perform basic actions with computers and other means of information and communication technologies based on familiarity with a personal computer (personal computer device) as a technical means.
- Use ergonomic methods of working with a computer and other means of information and communication technologies that are safe for the organs of vision, nervous system, musculoskeletal system; perform compensating physical exercises (mini-charging).
- Understand the composition of the computer and the purpose of its main devices for input, output, transmission and processing of information, know about different types of computers and digital devices for different types of work with information.
- Know about the software and its purpose, perform basic actions with the program menu, commands of the digital object management interface on the computer.
- Organize the storage of your own information in folders and files on your computer and external digital media.
- Understand the need to apply health-safe methods of working with computer devices in life, in e-learning, digital educational environment.
- To apply in life the ways of storing your own information when organizing a personal information space, a personal account, an electronic diary.
- Build a plan for the implementation of specified actions to solve a learning task using a computer and tools of simple programs.
- Understand and execute a simple algorithm for controlling the executor of commands using digital devices.
- Use a computer to solve accessible learning tasks with simple information objects (text, drawings, available electronic resources).
- To select the result of video recording and photographing suitable in terms of content and technical quality, use removable media (flash cards).
- Describe an object or process of observation according to a certain algorithm, record audio-visual and numerical information about it using information and communication technology tools.

- Collect numerical data in natural science observations and experiments using digital devices, a camera, a microphone and other means of information and communication technologies, as well as during a survey of people.

Communication literacy. Section "Digital Communications". The student will learn:

- Use a computer to search and reproduce the necessary information.
- Search for information in age-appropriate digital dictionaries and reference books, electronic educational resources, controlled Internet, a search system inside a computer; compile a list of information sources used (including using links).
- Use means of communication on the example of a resource of an educational organization (e-mail, school website, means of interaction in a group on a school website).
- Understand the importance of anti-virus protection of a computer, assigning a password, rules of speech etiquette when working with means of communication on the Internet.
- Create text messages using information and communication technology tools, edit, format and send them by e-mail.
- Create simple messages in the form of audio and video clips or slide sequences using illustrations, video, sound, text; post a message in the information educational environment of an educational organization.

3.3. *Additional Digital Competencies in An Interdisciplinary Technologies*

Individual tracks of children in creativity with information and communication technologies are:

- Competently use a computer keyboard, use computer translation of individual words, use available techniques for working with ready-made text, visual, audio information on the Internet, available ways of obtaining, storing, processing it.; use basic means of digital communications; record the progress and results of communication on the screen and in files; competently formulate queries when searching on the Internet, evaluate, interpret and save the information found; be critical of information and the choice of the source of information, participate in collective communicative activities in the information educational environment; observe the norms of speech interaction in interactive communication: e-mail, Internet, video services and other types and methods of communication; (philological cycle).
- Present data by various means of information and communication technologies, use information and communication technology tools (photo and video camera, microphone, etc.) to record and process information, prepare small presentations based on the results of observations and experiments; (mathematical and natural science cycle).
- Perform simple drawings and ornamental compositions using a computer graphics environment; get acquainted with a graphic tablet, touch screen, voice input;

print, scan drawings and texts, use a scanned text recognition program in Russian, design sound and music fragments using a computer and a musical keyboard, collect music collections, music library, video library; (cycle of art objects).

4. The Content of the Course "Digital Literacy" in Primary School

Section 1. "Working with digital information" (30 hours).

1.1. **Information.**

Man and the digital environment. Digital information and a computer.

Representation of digital information, the concept of digital data. Information, its collection, analysis and systematization. Types of information: text, graphics, numbers, multimedia, sound and video.

The concept of information processes of processing, searching, transmitting, collecting, storing information.

The description of the simplest information model is a table and a diagram.

Project. Interpretation of table data. Reading and filling in the table. Graphical representation of numerical data in a table. Reading a bar chart or pie chart.

1.2. **Processing of information on a computer.**

Information and computer. Working with simple information objects (text, table, drawing) on a computer in text and graphic editors: transformation, creation, saving, deletion.

Computer presentation. Preparing text, working with tables, inserting graphics and videos, recording audio.

Project. E-book (text, illustrations, audiobook, hypertext).

Project. Presentation on the topic of the project. A media book based on a presentation.

1.3. **Action planning and management**.

Using a computer for calculations. The order of actions. Calculation algorithm on a software calculator.

Team. The executor of the commands. The algorithm for managing the performer. An algorithm with a choice of action. The algorithm with the repetition of commands.

Project. Computer environment for managing the performer. Implementation of the algorithm on a computer in a learning programming environment.

Project. Software-controlled devices (control programs for household appliances, timer/alarm clock in a digital device).

Section 2. "Computer practice. Digital technology" (20 hours).

2.1. **Familiarity with the means of information and communication technologies, hygiene of working with a computer.**

Man and computer. Compliance with safe methods of working on a computer; careful attitude to electronic technical devices.

The diversity of computers and the world of professions using information and communication technologies.

The purpose of the main computer devices for input, output, and information processing. Turning on and off the computer and the devices connected to it. Computer programs.

Graphical computer management interface. Programs. Menu

Files and folders, their storage system on the computer and external devices.

Project. Ergonomic methods of working with a computer and other means information and communication technologies and physical exercises (mini-charging).

Project. Digital sensors, digital laboratories and devices for research and recording observations.

2.2. **Input and output of information: text, sound, image, digital data.**

Keyboard, a general idea of the rules of keyboard writing, using a mouse to work on a computer screen, using the simplest tools of software editors. Data output to the printer. Information in the form of photos, audio and video fragments. Microphone and headphones. Photo and video camera.

Project. Graphic tablet, touch screen, sound synthesizer, document camera, video camera in educational creative activity.

Project. Digital devices for people with disabilities.

Section 3. "Digital communications" (20 hours).

3.1. **Computer network Internet.**

Man and society. The Internet. Global computer networks. Website. Rules for a secure Internet connection, password and personal data.

3.2. **Transmission and retrieval of information.**

Means of communication: mobile telephony, e-mail, audio and video services. Antivirus protection of the computer.

Mass media: radio, television, press on the Internet. Selectivity in the use of mass media in order to preserve spiritual and moral health.

3.3. **Working with digital communications.**

Digital communication in education. Digital educational environment resources, registration, personal account, electronic diary, electronic portfolio, educational organization website, class forum, electronic reception.

Digital communications in life. Personal and collective network services: messengers, blogs, email, social network page. Network etiquette in the public space.

Project. Work with the means of communication – e-mail, educational sites on the Internet. Search for information. "Smart" devices connected to the Internet (Internet of Things).

Project. Settings tools for viewing websites for people with disabilities.

Topics of digital projects for the organization of creative interdisciplinary work in groups of students' choice.

Work with digital educational resources, ready-made materials on electronic media. Audiobooks. E-books and textbooks. Electronic music collections. Video materials. Creating a directory of links of additional cognitive information resources to the topic of the lesson.

Observations, recording and processing of the results of observations (nature, weather) and surveys by means of information and communication technologies. Presentation. Report. Reportage.

The use of various artistic techniques in individual and collective activities by means of information and communication technologies, computer animation, video and photo shooting, computer graphics. Media book. Animation. Exhibition.

Search and listen to music from electronic collections. Creation of information support for a music project (poster, presentation, invitation cards, etc.).

5. Educational Activities of Children Based on Digital Competencies

For the formation of initial digital literacy, the content of the training course is implemented using practical tasks with information and communication technologies in a group and individually, in partnership with a teacher. Practical work using a computer, presentation equipment, devices connected to a computer, as well as digital educational resources on the topics of the course (media studies) in the sections of the course content include the following types of practical information activities of children.

Working with digital information. Working with digital information objects that combine text, drawings, diagrams, tables, visual and graphic images (diagrams) and numerical data, fixed and moving images, sound, links and multimedia objects, digital objects for programming their management, which can be created stored on a computer and external media or posted on the Internet, transmitted via digital communications.

Practice working on a computer. Digital technology. Work with diagrams, descriptions and instructions, participation in explaining the purpose of various computer devices, digital environment devices, functions of household appliances with software control. Working with a computer and digital devices connected to it. The use of generally accepted programs in educational tasks for simple processing, storing information using information communication technologies: interface with the device, files and folders, selection of tools for working on a computer, input, editing of various types of information on a computer: text, sound, image, digital data; presentations. General safe and ergonomic principles of working with computers, digital devices and programs.

Digital Communications. Use e-mail, computer network, educational and cognitive sites in information and educational activities. To search for and transmit information, to assess the need for additional information for solving educational tasks and inde-

pendent cognitive activity; to identify possible sources of its receipt; to be critical of information and the choice of the source of information, to comply with the norms of safe behavior in the digital environment and the protection of personal data in personal network services.

6. Conclusion

The application of digital skills in practice in the subject education of younger schoolchildren includes regular practical work of children based on the use of information and communication technologies in all academic subjects and in the form of interdisciplinary educational projects using digital tools and devices.

Types of interdisciplinary projects:

Subjects "Mathematics and Computer science".

- Projects related to the presentation, analysis and interpretation of data; the ability to extract the necessary data from tables and diagrams, fill out ready-made forms, explain, compare and summarize information, draw conclusions and forecasts.
- Projects for managing the work of the team executor, programming algorithmic tasks and designing information objects.

Subjects "Language. Literary reading. Native language and literature. A foreign language".

- Projects in oral and written communication using information and communication technologies in order to present, formalize the results of the project in creative groups and search for the necessary information in various sources to complete educational tasks.

The subject "The surrounding world".

- Projects for working with information and communication technologies-means, searching for information in electronic sources and the controlled Internet to create messages in the form of texts, audio and video fragments, presentations based on the results of observations and research.
- Projects for the development of digital laboratories.

The subject "Technology".

- Practical work with a personal computer as a technical means, with its main devices for their intended purpose; experience of group and individual work with simple information objects: text, drawing, audio and video fragments; techniques for searching and using available electronic resources.
- Robotics projects.

Subjects "Art".

- Projects for the use of various information and communication technology tools in creativity (graphic editors, multimedia, augmented reality, virtual reality).

The subject “Physical culture and sport”.

- Projects in support of health-saving technologies of information activities, independent planning and performing physical exercises while working on a computer.
- Projects for the development of the simplest digital health sensors.

The course is implemented using the author’s textbooks (Tsvetkova *et al.*, 2022) and books to them.

Methodological materials, work program and electronic educational resources for the training course are available on the website of the author’s workshop for textbooks (BINOM, 2022).

Additional materials on project activities can be found in publications on media education (Bondarenko *et al.*, 2018, 2020).

Reference

FLERF (2012). *Federal Law of Education in Russian Federation*, No. 273-FZ of December 29, 2012. On Education in the Russian Federation. https://zakon-ob-obrazovanii.ru/

MERF (2021). *Ministry of Education of the Russian Federation*. Order No. 286 dated May 31, 2021. Federal State Educational Standard of Primary general Education. https://fgosreestr.ru/educational_standard/federalnyi-gosudarstvennyi-obrazovatelnyi-standart-nachalnogo-obshchego-obrazovaniia

Fgosteestr (2022). *Program of primary education of the Ministry of Education of the Russian Federation*. https://fgosreestr.ru/poop/primernaia-osnovnaia-obrazovatelnaia-programma-nachalnogo-obshchego-obrazovaniia-1

GRF (2021). Strategic directions in the field of digital transformation of education related to the sphere of activity of the Ministry of Education of the Russian Federation. *Decree of the Government of the Russian Federation* No. 3427-r of December 2, 2021.

Fgosteestr (2020)**.** The approximate educational program of the course “Information security” for primary general education. https://fgosreestr.ru/oop/informacionnaya-bezopasnost-1-4

Tsvetkova M. *et al.* (2022). *Informatics (in 2 parts). 3rd grade. 4rd grade*. Prosveshenie Publishing House, Moscow. (In Russian: Могилев А.В., Могилева В.Н., Цветкова М.С.. *Учебник: Информатика в 2 частях. 3 класс. 4 класс*. Издательство БИНОМ, Издательство Просвещение. Москва). https://lbz.ru/books/750/

BINOM (2022). *Methodical website of BINOM publishing house*. Author’s workshop. https://lbz.ru/metodist/authors/informatika/5/

Bondarenko E.A. *et al.* (2018)**.** The introduction of media education into the practice of general education. *Revista Espacios*, 39(38), 7.

Bondarenko E.A. *et al.* (2020)**.** *Development of Meta-Subject Skills of Schoolchildren in the Aspect of Media Education: Monograph*. Moscow, CJSC Parusa, 2020, 240 p.

M.S. Tsvetkova, professor of the Russian Academy of Natural Sciences, PhD in pedagogic science, prize-winner of competition "The Teacher of Year of Moscow" (1998). from 2002 to 2018 she is a member of the Central methodical commission of the Russian Olympiad in informatics and the pedagogic coach of the Russian team on the IOI. She is the author of many papers and books in Russia on the informatization of education and methods of development of talented students. She is the author of official textbooks and copybooks in Russia for primary school in Informatics. She is author and director of the International school in Informatic ISIJ (since 2017). She is the Russian team leader (2013–2017). She was awarded the President of Russia Gratitude for the success organizing the training of IOI medalists (2016). She is now the Expert of Committee on Education and Science State Duma of the Russian Federation (since 2017), and she has the Committee on Education and Science State Duma Gratitude (2021).

E.A. Bondarenko, Associate Professor of the All-Russian State Institute of Cinematography named after S. Gerasimov, PhD in pedagogic science. From 2001 to 2013 – Head of the Media Education Laboratory of the Institute of Content and Teaching Methods of the Russian Academy of Education, since 2014 – President of the Association of Film Education and Media Pedagogy of Russia. She is the author of many books and papers in Russia on the problems of film education, media education, information literacy, education standards, and information security. Expert of the Book Chamber (examination of textbooks), member of the jury of more than 20 contests of school media creation and film festivals.

I.Yu. Khlobystova is an Assistant Professor, PhD in pedagogic science at Glazovsky State Pedagogical Institute named after V. G. Korolenko, Glazov, Russian Federation. Her research interests include programming and computer science education. She is the author of many articles in Russia on teaching computer science at school and university. She has many years of experience in online publishing and working in editorial boards, including in publications related to the formation of digital media literacy. She is the author of official textbooks and copybooks in Russia for schools and colleges in Informatics. Expert of the Expert Council on a Secure Information Environment.

E.V. Yakushina, Moscow City University, Russian Federation, senior researcher of activity education laboratory PhD in pedagogic science. From 1996 to February 2015, she worked as a senior researcher at the Laboratory of Media Education at the Institute of Content and Teaching Methods of the RAO. Since 2014 he has been working as a publishing editor of the magazine «Media. Information. Communication». He has many years of experience in online publishing and working in editorial boards, including in publications related to the formation of digital media literacy. She has experience of participating in various research and expert projects conducted by the Ministry of Education and Science of the Russian Federation from 2000 to 2021, including work as a researcher in the Research Group on the Inclusion of Humanities Scientists in a new communication Environment (Institute of Psychology of the Russian Academy of Sciences). Member of the Expert Council on a Secure Information Environment.

Olympiads in Informatics, 2022, Vol. 16, 173–195
© 2022 *IOI, Vilnius University*
DOI: 10.15388/ioi.2022.14

Hosting IOI 2019 Azerbaijan: Back to the Future

Araz YUSUBOV[1], Farid AHMADLI[2], Jamaladdin HASANOV[1]
[1]*School of IT and Engineering, ADA University, Baku, Azerbaijan*
[2]*Amazon, Vancouver, Canada*
e-mail: ayusubov@ada.edu.az, faridra@amazon.com, jhasanov@ada.edu.az

Abstract. Who would guess that the 31st International Olympiad in Informatics (IOI) in Baku, Azerbaijan will be referred to as "the last normal" one for so long? While the IOI community looks forward to the upcoming onsite events, the readers may be interested in a retrospective report on the IOI 2019 from the host organizing, scientific and technical committees. This report covers the whole process throughout bidding, preparations, event days and closeout, shares some pain points and provides a number of practical recommendations. The authors hope that by looking ***back*** at the IOI 2019 Azerbaijan they will also help potential hosts of ***future*** IOIs.

Keywords: IOI, Azerbaijan, organizing committee, scientific committee, technical committee, CMS, project management.

Introduction

Usually all IOI host countries publish reports on the organizational aspects of the Olympiads in informatics, including the international ones (Iglikov *et al.*, 2013; Abam *et al.*, 2017). The goal of this report is to highlight some critical points of organizing an IOI specifically, while focusing on three aspects from organizational, scientific and technical perspectives. This report may also be useful for the future IOI hosts as it walks through the whole process starting from bidding and ending with the event itself.

1. *The Way to IOI 2019*

Before starting any enterprise there is that 'why?' question, the rationale for starting this endeavor. IOI is a massive endeavor, indeed, a week-long event that apart from the scientific component of preparing world-class original and challenging programming problems, as well as technical component of running a distributed computing environment with hundreds of workstations, also includes a heavy logistical component of accommodating hundreds of majority non-adult people, who travel from virtually all around the world. Plus, practically all of it is at the host's expense.

Unlike many other international events, such as popular sportive competitions, it would be naive to consider IOI as a means of promoting the host country as a tourist destination. In many countries IOI is virtually unknown (unfortunately) outside of the professional community of interest. However, as one of the internationally recognized science olympiads and the second largest one, initiated by UNESCO back in 1989, it is considered as "one of the most prestigious computer science competitions in the world" (Wikipedia, 2022).

1.1. *Initiation of Bidding*

A famous Chinese proverb ascribed to Laozi (Wikiquote, 2022b) says "a journey of a thousand miles begins with a single step." As per the official Regulations (IOI, 2022) (statute S4.1), process starts with a Letter of Intent sent by an official representative of an interested IOI member Country to the IOI Secretary. Becoming an IOI member assumes, among other requirements, a demonstrated capability of selecting a National Delegation through running national level selection of Contestants, and ability to sustainably participate in the last three IOI's after visiting as an Invited Observer for one year.

Azerbaijan joined IOI in 1994 and has been participating uninterruptedly in every IOI since then. For many countries the participation becomes possible thanks to enthusiastic professionals and volunteers, who apart from running the local selection of contestants, over the years put a lot of effort into resolving logistical issues such as securing sponsorship for the travel of delegations. In Azerbaijan, Dr. Ramin Mahmudzade (IOI, 2020) has been leading these efforts as also a Delegation Leader at 21 olympiads between 1996–2019. A decorated educator and well respected informatics professor, he was the main initiator and promoter of Azerbaijan hosting an IOI.

In Project Management terms, some differentiate the project initiator, as an individual, who promotes the project and champions its initiation, and the project sponsor(s), as one or more individuals or organizations that provide resources and support for the project and are accountable for enabling success (PMI, 2022). The Letter of Intent for hosting IOI in a specific year usually is sent to the Secretary of the IOI by the primary sponsor organization and provides reference to previous experience of hosting similar events as well as commitment to host the IOI up to the required standards. Upon receiving acknowledgement and rules from the Secretary the Country receives a Potential Host status.

The next step for the Potential Host is to present its bid to the International Committee (IC) of the IOI in the form of a report, which would include some details e.g. draft program and proposed venues, consolidated technical and human resources, draft budget and sponsorship plan, etc. Based on these reports from Potential Hosts the IC nominates a single candidate for hosting the IOI for a specific year, to be ratified by the General Assembly (GA) voting. Upon ratification, the Country receives Candidate Host

status with an official Invitation for hosting the IOI in that year. Azerbaijan presented its report for hosting IOI 2017 during the IOI 2013 in Brisbane, Australia, and later for hosting IOI 2019 during the IOI 2015 in Almaty, Kazakhstan. A positive constructive feedback received from the IC members after the first presentation helped with Azerbaijan's second successful bidding.

In Azerbaijan, the Ministry of Education was the championing government agency for both bids, while ADA University was the championing educational institution for the IOI 2019. Both organizations responded to the Invitation with an official confirmation upon which Azerbaijan became the Future Host for IOI 2019.

1.2. *IOI is the Way*

"There is no way to IOI, IOI is the way." This quote (homage to A.J. Muste (Wikiquote, 2022a)), which comes from the IOI 2019 General Assembly (GA) closing remarks, reflects the transformational role of IOI in the countries that join it and host it, hence may serve as a reply to our initial 'why?' question. In Azerbaijan, years ahead of the IOI 2019, starting from 2015 when the country became Future Host, were marked by increased attention and national level support to the competition and informatics in general, which eventually resulted in medals (IOI, 2021) after almost a decade-long gap and long-awaited announcement of including informatics questions in the centralized university entrance exams.

1.3. *You Won the Bid: Now What?*

The journey just starts with an IOI Country winning the bid and receiving the Future Host status. By that time usually there is already an official local Organizing Committee, which is to consolidate resources for arranging the IOI they won the bid for.

Three representatives of the Future Host automatically become members of each of the three long-term standing IOI committees, that is IC, the International Scientific Committee (ISC) and the International Technical Committee (ITC). This regulation ensures that the Future Host develops a better understanding of the corresponding aspects of organizing another IOI. In addition, IOI Regulations make it mandatory to invite a limited number of observers representing the Future Host to each IOI, which also facilitates further knowledge transfer. For this reason also, usually, the Future Host delegation includes many additional guest members, especially the year before the IOI they will host.

The authors discuss three aspects of running an IOI from organizational, scientific and technical perspectives, as they resided in the abovementioned committees and led the local efforts in corresponding three directions within the host Organizing Committee for IOI 2019.

2. General Organizational Matters

Every IOI is a unique event with a specific technical content and international nature, which often requires state-level efforts. Hence, having government agencies as part of the Organizing Committee is not merely a matter of prestige, their direct support is crucial for many activities, such as securing visa arrangements, safety and security, public media promotion, and access to some resources.

It is critical to build effective organizational structure and internal communications, way before any event activities commence. Project sponsors form the core of the host Organizing Committee and usually senior representatives of these organizations become members of the Steering Committee, which monitors the progress of preparations and provides support in consolidating necessary resources.

In Azerbaijan, the Steering Committee members included then head of the championing government agency, Ministry of Education, Mr. Jeyhun Bayramov, acting as the official Chair of IOI 2019, as well as the Minister of Transport, Communications and High Technologies as the Co-Chair. Finally, the true champion of the project and driving force of the committee was the Honorary Co-Chair of IOI 2019, Ambassador Prof. Hafiz Pashayev, Rector of ADA University.

The first semi-formal meeting of the Steering Committee was held as early as in March 2018 with a total of two working meetings in 2018 and four meetings held in 2019 with their participation. Having high-level decision-makers on board and effective government support helps to quickly resolve many administrative or logistical issues. In Azerbaijan, ultimately an official government-level organizing committee for IOI 2019 was formed by an order signed by the Prime Minister (ICT, 2019) in May 2019.

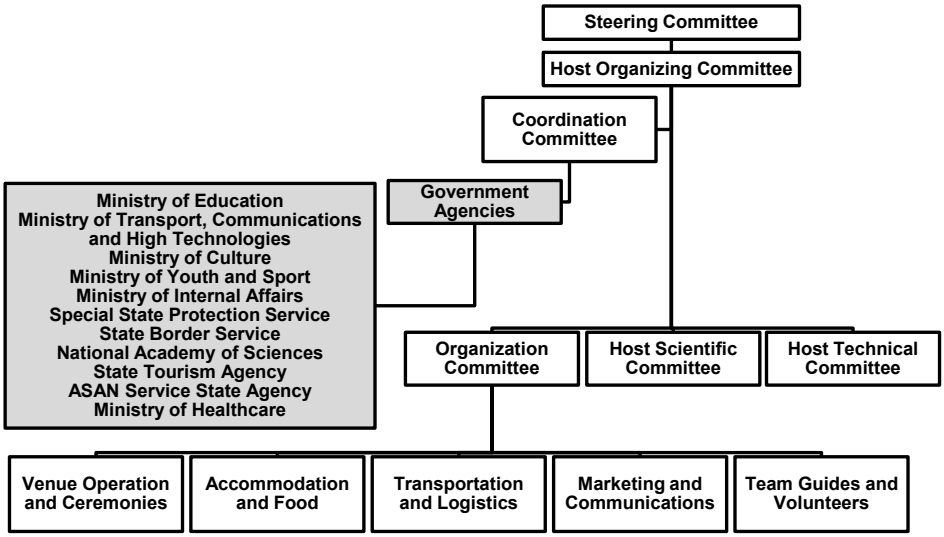


Fig. 1. Final organizational structure for IOI 2019 in May 2019.

Since the scope of an IOI includes a considerable element of event logistics, usually an effective organizational structure also includes a major event management vendor. In Azerbaijan, this role was filled by Caspian Event Organisers (CEO) company, which earned its reputation over more than two decades as an organizer of major international exhibitions or events, such as annual BakuTel Azerbaijan International Telecommunications, Innovations and High Technologies Exhibition (Bakutel, 2022) or Asian Development Bank Annual Meeting (ADB, 2015) in Azerbaijan. It is a good idea to include representatives of this vendor as part of the National Delegation to the previous IOIs.

To ensure effective internal communication and coordination, five working groups for Venue Operation and Ceremonies, Accommodation and Food, Transportation and Logistics, Marketing and Communications, Team Guides and Volunteers were established under the Organizing Committee (see Fig. 1) with relevant members both from ADA University and CEO.

It is also critical to build effective public communications months and even years ahead of the event. Involvement of the IOI committee members in local promotion helps emphasize the international importance of the event and gain further local support. We would like to thank the IC members, Mr Eljakim Schrijvers and then IOI President Prof Krassimir Manev for accepting the invitation to visit Baku in December 2016 and for participating in public discussions, interviews, as well as official meetings with government representatives. Prof. Manev's participation as one of the original organizers of the first ever IOI added a distinctive weight and energy to the local promotion.

A major milestone ahead of the IOI is a special meeting of local and international committees to be organized about six months before the olympiad by the Present Host. The primary goal of this usually three-day event, also known as Winter Meeting, is to jointly examine 'on the spot' the organization of the upcoming IOI. It is also a good opportunity for further public promotion. In Azerbaijan, the meeting dates from 18–20 February were used to organize a press conference and interviews (IOI Azerbaijan, 2019).

2.1. *Going Forward with the Project*

As in any project, for hosting IOI it is important to clearly define the scope, firmly secure the budget and get final agreement on the schedule. There are efforts to institutionalize the knowledge transfer within the IOI committees. For example, IOI Secretariat archives a number of 'checklist' documents that list critical activities pertaining to hosting an IOI. These would help with better understanding of the scope of the work.

We would also like to acknowledge with gratitude the IOI 2018 Japan team for producing an internal Final Report document that includes many useful details i.e. organizational structure, design and color coding of official badges, detailed programs including opening and closing ceremonies, venue maps and seating charts, schedules for daily operations and volunteer shifts, emergency procedures and more. It was used as a model, while IOI 2019 produced a similar document for internal use by Future Hosts.

In Project Management terms (PMI, 2022), Work Breakdown Structure (WBS) is "a hierarchical decomposition of the total scope of work". It is later translated into schedules with project activities, and budgets with projected costs and assigned resources.

As per an agreed model, the major services were procured by ADA University, while the CEO company was contracted to control the total budget for procurement of numerous other, relatively smaller services (Fig. 2).

In Azerbaijan, the Ministry of Education ensured allocation of the budget for IOI 2019. Such a model with a single source of funding is very helpful if there are solid guarantees of support. In an alternative model where there are several sources of funding at different levels (e.g. diamond, gold, silver, bronze), a separate organizing committee unit would manage "relationships with co-organizers and sponsors". For example, the final list of sponsors for IOI 2018 Japan (IOI, 2018) included around 50 organizations.

On a global level, from 2017 to 2021, IOI also had Acer as Official Sponsor (Acer, 2019), who committed to "supply notebooks for contestants and staff, and servers to run contest management systems and language translations". This arrangement helped the organizing committees both from the budget and scope perspective. Special thanks go

1. Venue renting (ADA)
   - 1.1. National Gymnastics Arena
   - 1.2. Boulevard Hotel
   - 1.3. Opening and closing venues
2. Accommodation (ADA)
   - 2.1. Boulevard Hotel (leaders, guests, organizers, volunteers)
   - 2.2. Athletes Village (contestants, volunteers)
3. Furniture and equipment (CEO)
4. Catering
   - 4.1. Leaders and organizers (ADA)
   - 4.2. Guests (ADA)
   - 4.3. Contestants (ADA)
   - 4.4. Volunteers (CEO)
5. Transportation (ADA)
6. Outdoor promotion (CEO)
7. Swag, souvenirs and awards (CEO)
8. Registration (CEO)
9. Printed materials (CEO)
10. Excursions (CEO)
11. Consultancy (ADA)
12. Video and photo shooting (CEO)
13. Musical program (CEO)
14. Website and social media development and maintenance (CEO)
15. Contingency costs
    - 15.1. Contingency costs (ADA)
    - 15.2. Contingency costs (CEO)

Fig. 2. Top level budget elements for IOI 2019.

to Prof Greg Lee as the chair of the Acer sponsorship working group, who also served as IOI President from 2018 to 2021. In Azerbaijan, Acer also agreed to donate the notebooks used for competition as a legacy of IOI 2019 to be used for development of informatics education in the country (Acer, 2019). We would like to acknowledge the professionalism and responsiveness of Acer colleagues on local, regional and global headquarters level.

For many national level events, the schedules are also to be coordinated with government agencies to consider other important public activities. The IOI 2019 final dates of 4–11 August were agreed to fit between the 2019 Summer European Youth Olympic Festival (EYOF, 2019) and a public holiday. Knowing these dates earlier is important for National Delegations for planning ahead and coordinating, for example, with participation in other scientific olympiads.

Overall timeline can be divided to pre-IOI, IOI and post-IOI periods with the former further divided to pre-Winter Meeting and post-Winter Meeting periods. The latter is marked by increased intensity of activities that culminates with the launch of the event itself.

Table 1

Major milestones in the IOI 2019 timeline

Internal coordination	IOI operations	External communication
• 7 March 2018: Kick off meeting with Steering Committee		
	• 7 September 2018: Azerbaijan received the IOI flag at IOI 2018 in Japan	• 9 September 2018: Official Facebook page launched at `https://www.fb.com/ioi2019/` • 9 September 2018: Official website with call for tasks launched at www.`ioi2019`.az
• 18 October 2018: Official meeting with the Steering Committee		
• 20 February 2019: joint meeting of the Host Organizing Committee and International Committee	• 18–20 February 2019: IOI 2019 Winter Meeting	
	• 31 March 2019: National Olympiad in Informatics running on IOI infrastructure – Semi-finals	• 17 March 2019 – Official IOI 2019 logo design approved
• 16 April 2019: Contract signed with the approved vendor for event management	• 15 April 2019: Memorandum signed with ACER as part of the 5-year global partnership • 24 April 2019: Official IOI Registration System launched for IOI2019 at www.`ioiregistration`.org • 30 April 2019: Soft copies of official invitation letters to IOI Country delegations sent by emails	• 26 April 2019: Official website re-launched at www.`ioi2019`.az • 26 April 2019: Three official contact emails @ioi2019.az and inquiry handling procedures established

Continued on next page

Table 1 – continued from previous page

Internal coordination	IOI operations	External communication
• 15 May 2019: Order signed on establishing the government level Host Organizing Committee • 22 May 2019: Meeting of the Working Group of the Coordination Committee • 30 May 2019: Meeting of the Coordination Committee • 11 July 2019: Meeting of the Working Group of the Coordination Committee	• 5 May 2019: National Olympiad in Informatics – Finals • 4–11 August 2019: IOI 2019 held in Baku, Azerbaijan	• 5 May 2019: Official YouTube channel launched with the first video

2.2. *Looking Back at IOI 2019*

According to a post-event survey, IOI 2019 can be considered as another successful IOI thanks to the professionalism and dedication of many people across a number of organizations, including 148 team guides, lead guides and volunteers, who worked hard throughout the event week. There were 498 participants from 88 countries and regions, including 331 Contestants and 87 Delegation Leaders. In addition, 78 guests were hosted, including 7 juniors.

While internal communication with the event management vendor was critical, it took some 'warm up' time on the first day for it to settle. Establishing early on more detailed communication protocols and using Winter Meeting activities, as well as national olympiad semi-final and final rounds for testing them in practice and revising as needed could be helpful.

Another common pain point was that there were too many dependencies on a few people, for example the Organization Committee manager. Perhaps, further delegation of some roles would help in this situation.

Communication channels included also 3 official email addresses used for correspondence mainly with participants and occasionally with the general public. While the travel-related email was handled by the event management vendor, and the others by the ADA University, establishment of clear protocols and response templates was helpful during the peak times.

Every IOI is remembered by some local touch or novelty brought by the host. For example, as part of the registration, a reduced guest fee was introduced for an accompanying second guest, who agreed to share a standard room. As a result, sharing rooms among guests was not mandatory, rather encouraged in Baku.

The IOI 2019 brought all the contestants, team leaders and guests together for a Cultural Night organized along with an open-air dinner on day 6, also known as Excursion Day 2, where the teams were invited to demonstrate their talents and present their culture

Table 2

Statistics for IOI 2019 communication channels

	Facebook page	YouTube channel	Info email	Travel email	Registration email
by 11 July 2019	Posts: 77 Likes: 903	Videos: 16 Subscribers: 120 Views: 2,606	163	102	108
by 11 August 2019	Posts: 133 / 36* Likes: 2,382	Videos: 27 Subscribers: 506 Views: 38,343	1,190 / 84*	636 / 62*	449 / 28*

*during the event

through music, dance or some other performance on the stage. This tradition should continue as it very much speaks to one of the IOI goals (IOI, 2022) (statute S1.7), which is about "foster[ing] friendly international relationships among computer scientists and informatics educators."

A Book of Tasty Algorithms, with foreword by Prof Donald E. Knuth, a special edition of a book about Azerbaijani cuisine as a homage to the famous prologue to Knuth's Art of Computer Programming was presented as a gift to all IOI 2019 participants. A revered computer scientist, Prof Knuth was very kind to support this project and wrote an exquisite introduction. In addition, the participants had a chance to add a personal note to a special big folded card to be sent to the professor along with copies of the book. We were overwhelmed by a positive response from Prof Knuth himself: "The books arrived today, and I'm overjoyed to see that they were very attractively printed indeed. I was also quite touched (and "flabbergasted"!) by the one-of-a-kind thank-you sheet that was inscribed by so many participants of IOI2019. Wow! My wife – who is a designer – was also appreciative of the outstanding cover design, and the IOI logo." (IOI, 2019a)

An official postage stamp of the Republic of Azerbaijan featuring all logos of all IOIs starting from the very first IOI 1989 in Bulgaria was released on the stage during the Closing Ceremony and presented as a souvenir to all participants.

Finally, at the IOI 2019 Closing Ceremony, the official IOI flag has got a designated bag for transporting it from country to country.

3. Scientific Committee

The IOI 2019 Host Scientific Committee (HSC) was composed of members located in different countries. Team was managed and worked completely remotely, and except for the Winter Meeting and actual contest period, the team was regularly meeting online to discuss the state of preparations, to work on tasks, etc. The committee consisted of 15 members from Iran, Russia, Poland, Ukraine and Singapore.

HSC was divided into smaller groups, coordinated locally per country of residence. These groups were working independently on sets of tasks assigned. An experienced

team member was assigned as a coordinator to each group, which made the overall communication process much more manageable. Some members departed from the team after some period of inactivity, some joined the team as authors of the submitted tasks.

Establishing secure communication channels and uniform infrastructure for task preparations was challenging. It would help future IOI hosts a lot to have a standard cloud platform provided by IOI for the contest website, "call for tasks" submissions, task preparation/management, secure communication, worker/grader hosts for testing and contest simulations. It would especially help those countries, who would like to host the event, but lack any technical/scientific expertise to do so. Always relying just on community proves to be risky, hence, IOI budget could be put to a better use by providing at least some of these tools.

Overall task preparation process of the IOI 2019 can be evaluated as successful. According to the IOI survey, contestants were mostly satisfied with the difficulty and originality levels of the selected tasks and the quality of problem statements and test data.

However, the preparation process itself was not as smooth as it might seem. Due to lack of communication (it is hard to demand absolute commitment from remotely located volunteers) before the on site meetings, a lot of work had to be done on the spot. For instance, some test data for some contest tasks was under prepared, i.e., lacked deeper analysis on possible exploitation of test cases. Or another case, when some better solution for a task existed, found during the meeting, ruining initial sub task distribution. All such cases were resolved during the joint ISC-HSC meetings thanks to individual efforts and contributions of all committee members.

Initial task selection after the "call for tasks" was lacking. After short listing we were left with just several tasks, barely enough to cover the contest without any backups. Therefore, we had to rely on individual authors/committee members contributing to the set of tasks. Eventually, those authors were recruited to work in the HSC. Team had also to improvise on modifying existing task statements to maintain a balance between hard, medium and easy tasks for both contest days in the case some tasks are rejected during a GA meeting.

Call for tasks could be improved by introducing various rewards or benefits for selected task authors. Making those mandatory for host countries to compensate could be very helpful. Our decision to invite task authors, since they were also HSC/ISC members, was natural and seemed to be reasonable at the time. But initially, we had a hard time deciding on appropriate compensation for the authors who will not be able to attend the contest. Luckily everyone attended.

Last, but not least, after the last GA meeting/translation session, relocating to the contest area, preparing tasks statements for contestants, loading them into the system and proof checking everything needed a lot more time than initially anticipated. Translation session ending just a few hours before the start of the contest and having the actual contest area isolated and far away from the meeting venues had almost delayed the start time of the contest. Had it been the case, it would have negatively affected the overall contest experience. Thanks to overall team efforts we still managed to get everything done in a timely manner.

4. Technical Committee

The organization of such a wide-scale programming competition was quite a challenging task for the HTC as unlike previous hosts, Azerbaijan has never hosted regional or international programming contests before. The number of contestants in each stage of the national olympiad in informatics never exceeded 100, which would be hosted by several high school or university facilities, without extra complexity on the computational and network infrastructure.

Considering the high uncertainty of the project, HTC chose the systematic and formal management approach, which in turn delivered the results accepted by committee members. This report covers the management part of HTC. The analysis and recommendations based on contestant data analysis have been provided in a separate report published in 2021 (Hasanov *et al*., 2021).

As mentioned in the introduction part of the paper, the number of visiting countries and contestants was not much different from the previous years. There were no significant changes on the regulations part either (one of the mentionable changes would be removal of the Pascal language from the contest, which actually simplified things a bit). Considering the similar initial conditions and requirements, using best practices from the previous years and considering the lessons learned, was the right way to keep the direction (which wouldn't probably work both for the onsite and online contests).

Project Management Institute (PMI) defines a project as "a temporary endeavor undertaken to create a unique product, service, or result" (PMI, 2018). There is no doubt that this defines IOI as a project, as the "product, service or result" part of it is a of IOI activities that are planned during the project time. This part makes IOI different from the majority of the projects – the project team's job does not finish after the delivery of the final product (service or result part will be skipped hereafter), it just transforms from one stage to another. This second, short period process is not a project at all – it's called Operation (also used in conjunction with Delivery and Maintenance), which is run and regulated differently from the projects.

Considering these two stages, the corresponding formal approaches has been used:

1) For Project Management – PMI's Project Management guideline and standards.
2) For Operation Management – best practices, standards and guidelines of ITIL's Service Operation, Lean management and ISO9001 (Service Delivery).

This report briefly describes the implementation details of the mentioned standards, categorizing the experience as positive and unexpected.

4.1. *Positive Experience and Outcomes*

4.1.1. *Usage of Formal Project Management*

Those who follow the formal project management are steps ahead of those who try to run projects based on their experience or even worse, intuition – the PMI standard defines the knowledge areas, phases, processes and documents that are required in each phase

for the given knowledge area. Below is the list of knowledge areas and their correspondence with the IOI project:

Knowledge Area	Why it was important
Integration management	Definition of the project constraints (time, budget and scope). Formal Acceptance of the Project. Change Control Process.
Scope management	Requirement analysis, collection of the facts, previous experience. Building Work Breakdown Structure that depicts the scope of the work and also helps estimate the budget based on the listed items. The scope baseline also helps understand the number of resources required for the project.
Time management	Estimating the activity durations and their execution sequence. Since the event had a strict deadline, time was the main constraint. Time management techniques allowed us to properly order the execution of tasks.
Cost management	Estimating the cost for each activity and creating the budget forecast for the technical and organizational activities. The budget shall be carefully allocated in the right directions (OPEX and CAPEX) and distributed throughout the implementation period as planned.
Quality management	Building strategies on Quality Assurance and Quality Control. The scenarios for the inspection of the software, hardware and network solutions and prevention of the real-time problems have been prepared.
Human Resource management	The HTC team contained more than 60 people of different ages, skills and expectations. Knowledge of team acquisition and management helps acquire the required talents in a short period of time and move them smoothly through all the stages of team formation and performance. Understanding motivational theories helps putting the right person to the right job, using proper reward and recognition methods and keeping team spirit always high. The responsibilities of team members was built as a RACI (responsible, accountable, consulted, informed) matrix and shared with the corresponding teams.
Communications	The communication of the HTC chair starts long before the event and keeps going even long after the closing ceremony. Preparation and delivery or presentations, writing letters, e-mails, organizing phone/video calls, writing specifications, requirement analysis, meetings with vendors, suppliers, stakeholders and external and internal IOI team members requires good knowledge of formal/informal and verbal/written communication. Project Management standards state that communication is 90% of a project manager's job, which should be considered as a serious message: if you as the next host are deciding between two options for the HTC – a geek without communications skills and non-IT guy with good communication skills, you should definitely go for the second option.
Risk management	Unexpected things (good and bad, or positive and negative risks as we call them in Project Management) will definitely happen. The uncertain nature of projects may surprise with small or dramatic surprises (don't think that in 2019 Singapore team knew about the pandemic that totally changed the format).
Procurement management	Not all deliverables are going to be done internally. The majority of them, like venues, networking, power management, printers, transport, event management, catering, security will be rented or purchased. The Project Manager shall get familiarized with the procurement procedures (wish lists, PR/PO management, single sourced orders, tender rules and so on) of the leading organization and make necessary preparations.
Stakeholder management	IOI project involves almost 30 stakeholders (if not more) of different breeds, power and interest (probably should be published in a separate paper). Having this list helps in elicitation of requirements and not forgetting anyone or anything during the decisions and changes.

4.1.2. *Investigation of Previous Experience and Guidelines*

The IOI events hosted in previous three years were not the same in terms of technical setup and organizational details:

- In IOI 2016 (Kazan, Russia), as a contestant environment laptops were used. The scoring system was an in-house developed software called PCMS, a Windows-based system that is widely being used in Russia. The event was held on the campus of the Kazan Federal University.
- In IOI 2017 (Tehran, Iran), they used mini PCs with external monitors and keyboards and organized the contest on two floors of the same venue, located very close to the hotels where contestants, team and committee members have been staying. As a scoring system, the version 1.4 of CMS with local modifications has been used. By the way, with their new tools and automation systems, Iranian HTC made a tremendous contribution to IOI software and processes, which is still being used by the IOI community. Ansible has been used for the development of the automation scripts.
- In IOI 2018 (Tsukuba, Japan), the contestant machines changed back to laptops (provided by Acer as a part of sponsorship). The contest management system was CMS version 1.4 but with adapted modifications and improvements. Some of the tools developed in IOI 2017 have been adapted for the new procedures. Japanese HTC was the first who decided to run services in the cloud. There was a thoroughly designed model tailored for AWS, which was reviewed and accepted by ITC. The automation tools used in IOI 2018 were developed on Ruby based scripts.

Having such a variety of options actually was good for the analysis. It would help retrospectively evaluate each option and choose the best one. The collection of the information started in 2017, during Azerbaijani delegation's visit to Tehran for IOI 2017, where the discussions with Russian and Iranian HTC started. Both HTC chairs were kind enough to provide all the documents that they had about the preparation i.e. checklists, lessons learned, preparation plans and technical details.

Key takeaways from the previous IOIs were:

1) Having all the contestants in the same contest hall simplifies the administrative and technical tasks. Finding a venue with a given area, number of meeting rooms (for the committees), required lighting, entrance and WC for ~400 people and flexible infrastructure for the stage design is a challenging task. After evaluation of the previous options, it was decided to use an indoor sport hall. The best match was National Gymnastics Arena (NGA) in Baku, which:
    a) Was equipped technically (power and IT).
    b) Had enough meeting rooms.
    c) Had an Athlete's Village (AVL), where we decided to accommodate the contestants, on the other side of the road (there also was an underground crossing from the village to the NGA).
    d) Was in the perfect location – on the crossroads of the city entrance, right next to the metro and bus station and in a straight, 6–7 km way to the hotel (leaders' accommodation).

2) Accommodation of all the visitors is another challenging task – where to place ~700 people? Hotels usually do not have that many rooms. Even if they had, it would be quite expensive. Splitting people into two hotels would add extra administration and transportation costs. Considering this, we decided to split visitors into 2 groups:
   a) Team leaders, committee members and guests in a hotel.
   b) Contestants and their guides in AVL.

   By this solution we solved several problems at once:
   - Isolated contestants from the others not by the venue but by the distance too.
   - Managed to accommodate contestants from the same teams in the same apartments (with 3 or 4 bedrooms).
   - Accommodate contestants as close as possible to the contest hall.
   - Save on accommodation and transportation costs.

   The last item is very important – accommodation cost is the biggest piece in a cost distribution pie chart report.
3) Transportation is a bottleneck in all the processes. Arrival of the buses, their parking space, distribution of people and their gathering, driving time is always a waste that does not add a value to the process. We tried to eliminate this Muda (means "waste" in Japanese, from Lean Management) whenever possible. Our bigger achievement, again, was finding a contest hall and contestants' venue very close to each other. The local authority's recommendation on using transportation for the safety of contestants was successfully replaced with using a nice and even safer underground pass that connects AVL and NGA.

4.1.3. *Visiting Japan for the Observation and Participation in Technical Works*

After observing the process of IOI 2017 in Tehran and participating in the discussion as ITC members, it was decided to attend IOI 2018 before the contest and participate in the technical preparation works. IOI 2018 committee heads were very kind to accept this request and organized this process, such that our HTC chair arrived almost a week before the contest. The observation of the process was priceless. This is usually never shared with the committees that are mainly interested in the 'what (is done or remains)' question, rather than the 'how (have you done this)' question.

There were some modifications in Japanese implementation that made this process even more exciting (well, risky too):

- Using AWS infrastructure.
- The preparation of the contest hall started 3 days before the event date! Some of the previous HTCs would repeat that 2 weeks might not be enough for the contest hall works (network, furniture, setup, testing, etc.). Here we had only 3 days for it! Well, actually, they managed to finish it in 2 days, which convinced me of the power of organization and planning ("Flow" in Lean Production).

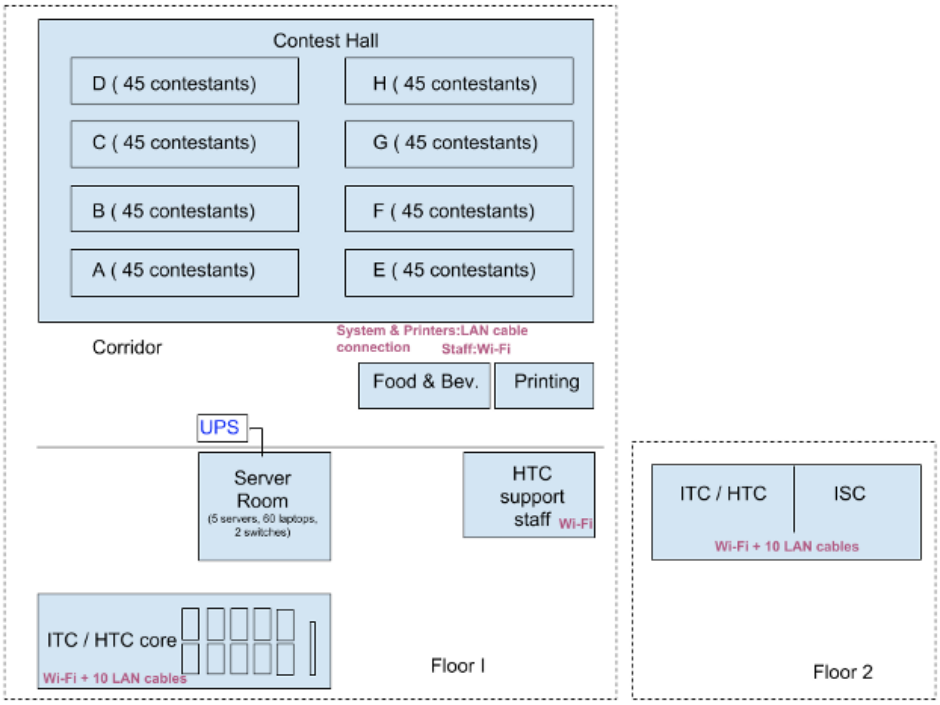


Fig. 3. The floors plan of the IOI 2019 contest hall in the National Gymnastics Arena.

These are the takeaways that we got from our observation:

1. From the floor planning, networking and troubleshooting point of view, indoor sports hall is the ideal venue for the contest. Division of the contest hall into 8 sectors (A–H) with 46 seats in each (a dedicated 48-port network switch per sector) was also adopted from the IOI 2018 design.
2. It is possible to set up a contest hall in 3 days. It requires good selection of reliable suppliers, good planning and coordination. In this case, the power engineers, network guys, furniture company and HTC team will work in an aligned way. It can be imagined as shifted signals i.e. when one line of desks are put, the network and power engineers would put cables, and HTC team guys put laptops on the desks and plug them to power cords and network cables. And meanwhile, the next row of the desks was placed next to each other.

   With this strategy we managed to finalize our setup in 3 days! It wasn't to beat the Japanese record, but having the same constraint. We had another event that finished a week earlier. Dismantling their stuff took exactly 3–4 days (In Tsukuba, they had a badminton contest that finished 4 days before the event).
3. In IOI 2018 the HTC was split into two parts and each of them had a coordinator: organization and technical. This isolation is a great idea, since problems during the real contest usually appear on both sides at the same time.
4. Japanese HTC used Slack for general messaging (different channels for different purposes) and push-to-talk for the organizational team. We adopted both

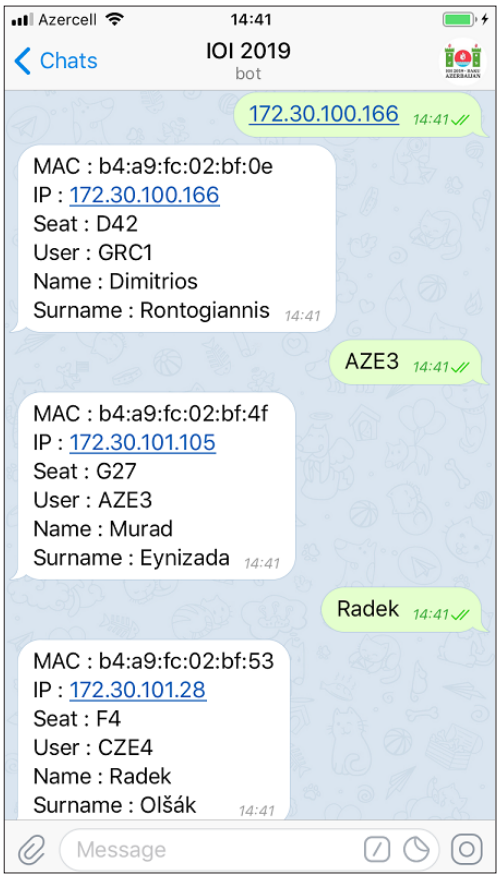


Fig. 4. The interface of the IOI 2019 Host Technical Committee telegram bot.

strategies but also added Telegram as an alternative. Telegram bot was used to locate the contestant by the IP, country, seat number or name (screenshot in Fig. 4 source code is shared in GitHub repository (IOI, 2019d)).

4.1.4. *Rehearsal*

Plan is a theoretical flow of the actions that considers the usual course of actions. It is always a good idea to run the action items in reality or at least simulate them. All the activities, from the hypothetical to routine works have been run in all the venues. Based on those experiments we adjusted our estimates on what is the actual time to set all the contestants at their desks, deliver a printed paper to a contestant, move from a server room to an IT meeting room, print translation copies for several countries and so on. Some of those experiments led to some changes in organizational setup. For example, it was realized that the food and beverage and the printing corners are not set optimally. Speaking of Lean principles, this is what is mentioned as “Form” (or “Kata” in Japanese).

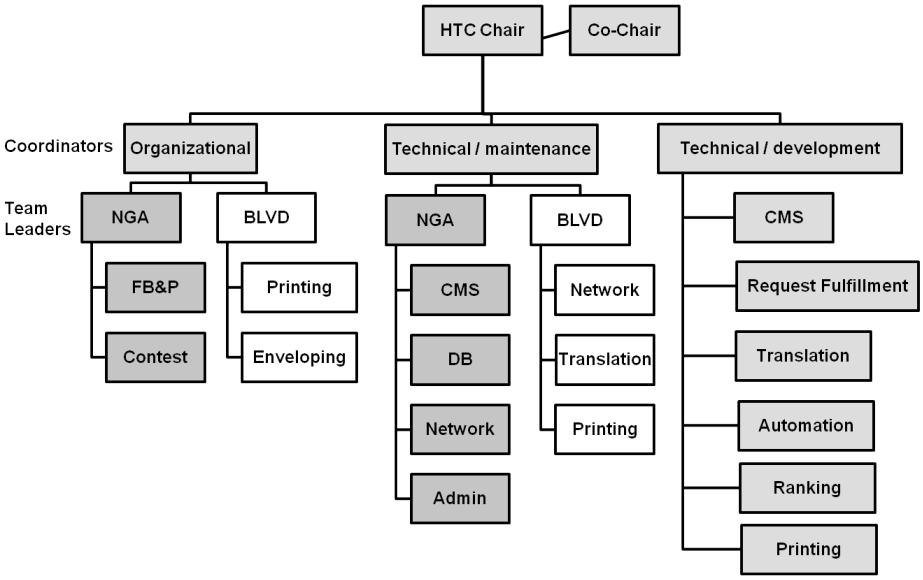


Fig. 5. Organizational structure of the IOI 2019 Host Technical Committee (NGA – National Gymnastics Arena, BLVRD – Boulevard Hotel).

4.1.5. *Utilization of University Resources*

The organization of an international event of such scale requires a decent workforce and other resources. When hosting organization is university, the majority of the required resources can be delivered by university, such as:

- Core HTC team. In IOI2019, the core HTC team had more than 60 people and almost all of them were from ADA University (3 faculty members, 3 people from IT department and others were undergrad students). We outsourced only the networking part. The structure of the HTC can be seen in Fig. 5.
- Procurement and Finance tasks had been done by the corresponding departments of the university.
- University's classroom and labs have been used for the meetings and contest environments.

4.1.6. *Formal Processes and Procedures*

With more than five functional teams (see Fig. 5) and numerous activities under each, asking everybody to memorize all the steps and conditionals wouldn't be the right choice. Additionally, acquiring new members and transferring members from one team to another is common throughout the process. Considering this, we decided to follow the ISO recommendations for the service quality: document what you do and do what is in the document. The majority of the technical processes are described in the "HTC Procedures" document located under the "Project docs" folder in (IOI, 2022). It helped us to speed up the onboarding process of new team members by just referring to the corresponding procedure.

4.1.7. *Acer Sponsorship*

Acer sponsorship is mentioned in the organization part. By having this opportunity we gained multiple benefits, putting cost savings aside:

- Had almost exactly the same contestant environment as in the previous IOI. We were familiar with user experience and common questions.
- Had a great technical support by the local Acer team. This sponsorship was not limited only to contestant laptops as all the grading machines and server equipment were from Acer, too. There was a case, when we needed to replace a network card – thanks to local Acer support we had it in NGA in less than half a day.

4.2. *Things that did not go as Expected*

There were some problems that affected the overall quality of the event, too. When analyzed, we can see that these are the tasks or processes that have been overlooked during the planning phase.

4.2.1. *The Network Structure wasn't Built in an Optimal Way*

The whole contest network has been designed as a ring i.e. from the system room switch to switch A (connecting contestant machines in section A), from switch A to switch B and so on. This creates serious problems during the mass updates and imaging. Ideally, it would be correct to run imaging for each section separately, which would be 8 times faster than we had. The main problem here was that the technical solution provided by the network supplier was not reviewed by the responsible member of the HTC team. Solution to this could be adding a checklist like "make sure the network design does not conflict with the HTC processes".

4.2.2. *Lack of Synchronization with the HSC*

Unlike HTC, our colleagues from HSC did not use formal communication or project management approaches. The outcomes of their discussions were usually documented as meeting notes or discussion logs. As HTC we have been constantly asking them to share the details on the given process.

As a result of such de-synchronization, two surprises arose: one during the day 2, when it was unexpectedly announced that the contestants needed some software for the visualization and another when the translation results ended much later than expected.

4.2.3. *Communication Problem with Vendors*

It's impossible for HTC to do all the work using its own organizational resources, as network equipment, furniture, printers and many other things are feasible to rent rather than to buy. For that very reason you need to work with suppliers. In our case, we had a

hard time convincing them all to use the standards applied by HTC or use formal project management. Each of them had their own understanding on projects, planning and standards. It would periodically bring surprises during the preparation phase. One of the critical cases was when the event organizer, responsible for printing the team/contestant labels used the old version of the file and as a result we received complaints during the practice day.

As a preventive action (of course for the future organizers) we recommend adding one more requirement to your RFPs: the project manager from their side needs to be formally certified in Project Management. Well, it might be disputed with a counter-argument that not all the certificated Project Managers are good ones, but you as an HTC chair will know that this person is going to speak the same language with you.

4.3. *New Translation Procedures*

The Translation Process is the most challenging one for HTC, with the following complexity factors:

- All the other committees are involved too – the process mainly led by HSC, regulated by ISC and IC and supported by HTC.
- Is the hardest process to plan and hence to automate – the flow of the process highly depends on participants and their discussions.
- Not limited in time from the organizational point but has to finish until the next contest – very short call for HSC and HTC.

Despite the mentioned challenges, there is a space for the automation of some part of the process. In 2017, Iranian HTC has put great efforts on optimizing and automation of the possible Translation processes (IOI, 2017):

- *Task Preparation System* has a nice web interface with the functionality of preparing the contest tasks and task statements.
- Translation System is a user-friendly markdown editing environment for translating the IOI tasks, with parallel view, PDF generation, notification system, and revision history.

In IOI 2018, Japanese HTC used the same tools but made modifications based on the previous comments and new requirements.

In IOI 2019, the Iranian version of the Translation system was used as a base for many core functionalities. Additionally some new features were added, such as:

- Monitoring of live translation and printing status of each team (can be projected on a big screen).
- Automatically mapping each student's preferred language (if provided early) to the printing system, so that it prints specifically in that language.
- Optionally, being able to print all the team tasks at once (merged printing), so that teams do not have to wait in lines for every student's copy.

4.4. *Summary*

Using previous experience will optimize and increase the quality of processes. The project documents, processes and all the other supplementary data used and generated by HTC can be found here (IOI, 2019d). Additionally, we would like to list some problems that would be good to fix for the next onsite events:

1. Minimization of the translation time. Long discussions and late contribution to the translation delays the printing and packaging process. Azerbaijani HTC raised a question of using the resources of IC, ITC and ISC to translate the tasks before the translation session. Back then there were committee members that spoke languages used by more than one country: Spanish, French, Russian, Arabic and Persian.
2. All our negative experiences are documented in the "Lessons Learned" document located in "Project docs" in (IOI, 2019d).
3. Printing of task descriptions seems redundant – it seems the contestants do not use them much. After the contests, we saw many unopened envelopes or opened envelopes with task descriptions inside. It could be surveyed and if 70% of the contestants do not need printed task descriptions, the printing process can be removed from the translation process. This would simplify and shorten the translation process. The students who need printed papers can print them on demand during the contest.
4. Some tools are required to be developed to improve the efficiency. Based on the previous experience, we decided to develop a custom application for the registration and fulfillment of the contestant requests. It helped to categorize, filter and measure the requests related data: each request had an assigned person (in a pull mode) which helped us to estimate the average resolution time for each type of request after 40 minutes of usage. A screenshot of this application is shown in Fig. 6.

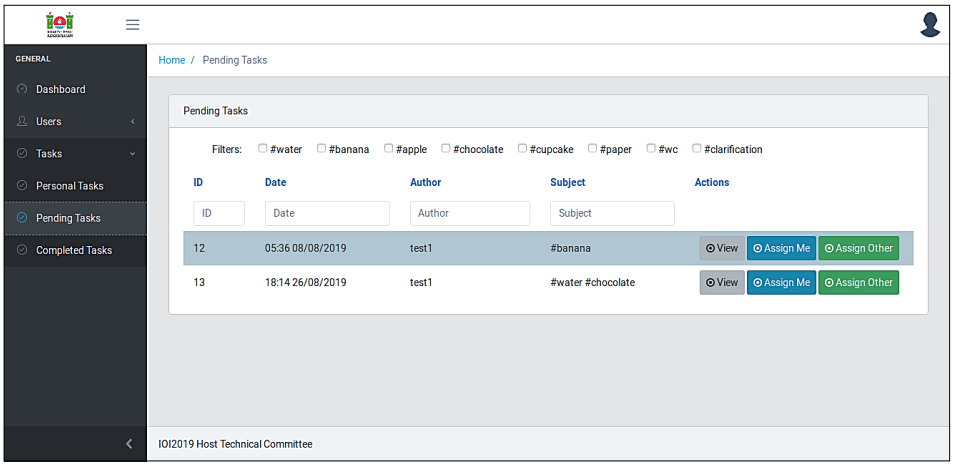


Fig. 6. The interface of the IOI 2019 request fulfillment tool.

Conclusion

The previous sections mention a number of recommendations based on the organizational, scientific and technical aspects of our IOI 2019 experience. Additional notes would be:

- Adapting the formal project management approach to organizing future IOIs with a regularly updated centralized repository of document templates will ensure both standardized knowledge transfer and increased quality.
- There is an educational component to be explored. For example, the experience of IOI 2019 was used as a case in IT Project Management (INFT 3609), as well as Systems Analysis and Design (INFT 2303) courses taught at ADA University.
- While the world is going through dramatic changes, we should be reminded that one of the IOI objectives (IOI, 2019b) is "to foster friendly international relationships among computer scientists and informatics educators." While putting the main emphasis on the competition component of the IOI, we should not lose the focus on all objectives. IOI Conference and Cultural Night initiative will gain additional meaning and importance.
- A centralized content management system for building official IOI websites would save host team's efforts for setting up this important communication channel, resolve the issue of archiving the historical content, and ensure a consistent look and feel.

Acknowledgements

The authors thank Emil Abbasov, member of the HTC team[1], for contributing to the paper with section 4.3 about new translation procedures.

Throughout the paper many colleagues have been acknowledged for their support in hosting IOI 2019. The authors thank all the people of IOI and feel privileged and humbled to be part of this amazing community. Special thanks go to Ramin Mahmudzade, Fuad Hajiyev, Krassimir Manev, Eljakim Schreivers, Bakhyt Matkarimov and all others who believed in and supported the vision for IOI 2019 in Baku, Azerbaijan. We thank all the IC, ISC, ITC members for their valuable inputs and recommendations.

HSC thanks all ISC members for the contribution into polishing the problem set; all invited HSC members, especially Bartosz Kostka, Danylo Mysak and Gleb Evstropov for their enthusiasm and cooperation; special thanks go to all Iranian colleagues, Kian Mirjalali and Ali Sharifi Zarchi (IC member) for volunteering to help and giving great support.

HTC team would like to thank: Acer's global and Azerbaijani team for their organizational and technical support; Sergei Masyagin (IOI 2016 HTC chair) and Hamid

[1] Doctoral student, School of Engineering and Applied Sciences, George Washington University, USA, eabbasov@gwu.edu

Zarrabi-Zadeh (IOI 2017 HTC chair) for sharing their experience and providing necessary documents; IOI 2018 IC chair Seiichi Tani and HTC chair Rie Yamaguchi for hosting and involving the HTC 2019 chair in their technical activities; William Di Luigi and Wael Eweda for their early arrival at IOI 2019 and exceptional support during the preparations and live sessions; The members of IOI 2020 Singapore HTC team, Ranald Lam, Lai Zit Seng and Lin Si Jie, for their involvement, acting as local HTC members, testing the infrastructure and providing support during the event.

References

Abam, M.A., Asadi, A., Ameli, A.J., Seddighin, S.R., Shahmohammadi, F. (2017). Iranian National Olympiad in Informatics. *Olympiads in Informatics*, 11(Special Issue), 25–33. https://ioinformatics.org/journal/v11si_2017_25_33.pdf

Acer (2019). Acer Joins the 2019 International Olympiad in Informatics (IOI) in Azerbaijan as Official Sponsor. https://news.acer.com/acer-joins-the-2019-international-olympiad-in-informatics-ioi-in-azerbaijan-as-official-sponsor

ADB (2015). *ADB Annual Meeting Sees Concrete Progress on 'Fostering Partnerships'*. Baku 2015. The Asian Development Bank (ADB). https://www.adb.org/annual-meeting/2015/main

Bakutel (2022). Azerbaijan International Telecommunications, Innovations and High Technologies Exhibition. https://bakutel.az/

EYOF (2019). The European Youth Olympic Festival – 2019 Summer Baku. https://www.eyof.org/information/

Hasanov, J., Gadirli, H. and Baghyev, A. (2021). On Using Real-Time and Post-Contest Data to Improve the Contest Organization, Technical/Scientific Procedures and Build an Efficient Contestant Preparation Strategy. *Olympiads in Informatics*, 15, 23–36. DOI: 10.15388/ioi.2021.03

ICT (2019). *The Order on Establishing the Organizing Committee for Arranging the International Olympiad in Informatics in 2019 in Baku*. Cabinet of Ministers of the Republic of Azerbaijan. https://ict.az/az/content/367/

Iglikov, A., Gamezardashvili, Z., Matkarimov, B. (2013). International Olympiads in Informatics in Kazakhstan. *Olympiads in Informatics*, Vol. 7, 153–162. https://ioinformatics.org/journal/INFOL124.pdf

IOI (2003). *IOI Regulations*. Approved by GA, August 2003: https://ioinformatics.org/files/regulations02.pdf

IOI (2017). Tools developed by Iraninan HTC. https://ioi2017.org/contest/technical/

IOI (2018). *IOI 2018: Japan. Tsukuba, Ibaraki*. https://ioi2018.jp/

IOI (2019a). *A Message from Prof. Knuth*. 31st International Olympiad in Informatics, 4–11 August, 2019, Baku, Azerbaijan. https://ioi2019.az/en-content-29.html

IOI (2019b). About IOI. 31st International Olympiad in Informatics, 4–11 August, 2019, Baku, Azerbaijan. https://ioi2019.az/en-content-3.html

IOI (2019c). IOI 2019 project documents. https://drive.google.com/drive/folders/1GvPj4zCqDtRaiQ6aTj6DMiKveocFbAwk?usp=sharing

IOI (2019d). Popular repositories. GitHub repository of IOI2019 Host Technical Committee. https://github.com/ioi-2019

IOI (2020). *IOI Statistics.* Azerbaijan – People – Ramin Alinazim Mahmudzade. http://stats.ioinformatics.org/people/3240

IOI (2021). *IOI Statistics.* Azerbaijan – Results. http://stats.ioinformatics.org/results/AZE

IOI (2022). *IOI Regulations.* https://ioinformatics.org/page/regulations/

IOI Azerbaijan (2019). *IOI 2019: Winter Meeting – Greg Lee*. Playlist https://www.youtube.com/watch?v=He2kF2UuSzA&list=PL8pL5Gws-n0zZEIIpxiAtl-BZdxLjUQi6

IOI Tools (n.d.). *Checklist*. "Technical guidelines for future hosts" by IOI International Technical Committee. https://ioi.github.io/checklist/

PMI (2018). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. 6th ed. Project Management Institute. https://book.akij.net/eBooks/2018/March/5abcc35b666f7/a%20guide%20to%20the%20project%20management%20body%20of%20knowledge%206e.pdf

PMI (2022). *PMI Lexicon of Project Management Terms : Version 3.2.*
https://www.pmi.org/pmbok-guide-standards/lexicon
Wikipedia (2022). International Olympiad in Informatics – Wikipedia. Last edited on 11 May 2022:
https://en.wikipedia.org/wiki/International_Olympiad_in_Informatics
Wikiquote (2022a). *Abraham Johannes Muste* (1885–1967). Last edited on 10 March 2022:
https://en.wikiquote.org/wiki/A._J._Muste
Wikiquote (2022b). 老子 Lǎozi (circa 6th–5th century BC). Last edited on 28 April 2022:
https://en.wikiquote.org/wiki/Laozi

A. Yusubov is an Assistant Professor of Computer and Information Sciences in the School of IT and Engineering at ADA University. Dr. Yusubov worked in academia, industry and international organizations, led and contributed to various educational projects, including the national FIRST LEGO League robotics tournaments for school children. He is ACM Senior Member and the founding member of the Azerbaijan ACM/ACM-W Chapter. Dr. Yusubov has been an IC member for the period of 2017–2020, elected again for 2021-2023, was local Host Coordination Committee Manager during IOI 2019.
http://stats.ioinformatics.org/people/6418

F. Ahmadli is a Software Development Engineer at Amazon office in Vancouver, Canada. Previously he was a Senior Instructor of Computer and Information Sciences in the School of IT and Engineering at ADA University. Mr. Ahmadli is an IOI veteran and competitive programming enthusiast. He has been an ISC member for the period of 2017–2020 and led HSC during IOI 2019.
http://stats.ioinformatics.org/people/750

J. Hasanov is an Assistant Professor of Computer and Information Sciences in the School of IT and Engineering at ADA University. Dr. Hasanov is mainly focused on image processing and machine learning problems covering text and digital object recognition domains. Additional to the research field, he teaches the management aspects of the IT in production and operation environments. Dr. Hasanov has been an ITC member for the period of 2017–2020 and led HTC during IOI 2019.
http://stats.ioinformatics.org/people/6162

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 16, 2022

Foreword 1

D. GINAT, S. ARIAN, O. BECKER
Posing Creative Reduction Tasks 3

A. KATYETOVA
How Competitions Can Motivate Children to Learn Programming 13

B. KOSTADINOV, I. STOJMENOVSKA
Common Approaches to Developing Extensible E-learning Systems 23

A. LAAKSONEN
What is the Competitive Programming Curriculum? 35

M. LANDMAN, G. FUTSCHEK, S. UNKOVIC, F. VOBORIL
Initial Learning of Textual Programming at School: Evolution of Outreach Activities 43

J. STAUB
Error Handling in XLogoOnline 55

M. S. TSVETKOVA, V.M. KIRYUKHIN, N. A. BORISOV, M.I. KINDER
Methods of Tracks for Training Juniors in Olympiad Informatics: The ISIJ Experience 75

R. WU, A. LV, Q. ZHAO
Detecting Plagiarism as Out-of-distribution Samples for Large-scale Programming Contests 89

REPORTS

M. DOLINSKY
Primary School Programming Olympiads in Gomel Region (Belarus) 107

P.S. PANKOV, K.A. URAIYMOV, A.A. BELYAEV
Olympiads in Informatics in Kyrgyzstan 125

Y. SAGYNTAY
Informatics Olympiads in Kazakhstan: Team Selection and National Olympiads in Informatics 135

D. TSEDEVSUREN, J. DASHDEMBEREL, T.-O. BATTOGTOKH, T. ULAMBAYAR, A. KHUDER
Organization and Results of Mongolian National Online Olympiads in Informatics 145

M.S. TSVETKOVA, E.A. BONDARENKO, I.Yu. KHLOBYSTOVA, E.V. YAKUSHINA
Digital Literacy in Primary School 159

A. YUSUBOV, F. AHMADLI, J. HASANOV
Hosting IOI 2019 Azerbaijan: Back to the Future 173

Publisher office: Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
August, 2022

Olympiads in Informatics

Volume 16, 2022

Foreword 1

D. GINAT, S. ARIAN, O. BECKER
Posing Creative Reduction Tasks 3

A. KATYETOVA
How Competitions Can Motivate Children to Learn Programming 13

B. KOSTADINOV, I. STOJMENOVSKA
Common Approaches to Developing Extensible E-learning Systems 23

A. LAAKSONEN
What is the Competitive Programming Curriculum? 35

M. LANDMAN, G. FUTSCHEK, S. UNKOVIC, F. VOBORIL
Initial Learning of Textual Programming at School: Evolution of Outreach Activities 43

J. STAUB
Error Handling in XLogoOnline 55

M. S. TSVETKOVA, V.M. KIRYUKHIN, N. A. BORISOV, M.I. KINDER
Methods of Tracks for Training Juniors in Olympiad Informatics: The ISIJ Experience 75

R. WU, A. LV, Q. ZHAO
Detecting Plagiarism as Out-of-distribution Samples for Large-scale Programming Contests 89

REPORTS

M. DOLINSKY
Primary School Programming Olympiads in Gomel Region (Belarus) 107

P.S. PANKOV, K.A. URAIYMOV, A.A. BELYAEV
Olympiads in Informatics in Kyrgyzstan 125

Y. SAGYNTAY
Informatics Olympiads in Kazakhstan: Team Selection and National Olympiads in Informatics 135

D. TSEDEVSUREN, J. DASHDEMBEREL, T.-O. BATTOGTOKH, T. ULAMBAYAR, A. KHUDER
Organization and Results of Mongolian National Online Olympiads in Informatics 145

M.S. TSVETKOVA, E.A. BONDARENKO, I.Yu. KHLOBYSTOVA, E.V. YAKUSHINA
Digital Literacy in Primary School 159

A. YUSUBOV, F. AHMADLI, J. HASANOV
Hosting IOI 2019 Azerbaijan: Back to the Future 173

ISSN 1822-7732

9 771822 773007