

Olympiads in Informatics

12

IOI
INTERNATIONAL OLYMPIAD IN INFORMATICS

ISSN 1822-7732

INTERNATIONAL OLYMPIAD IN INFORMATICS
VILNIUS UNIVERSITY

OLYMPIADS IN INFORMATICS

Volume 12 2018

Selected papers of
the International Conference joint with
the XXX International Olympiad in Informatics
Tsukuba, Japan, 1–8 September, 2018



OLYMPIADS IN INFORMATICS

Editor-in-Chief

Valentina Dagiene
Vilnius University, Lithuania, valentina.dagiene@mii.vu.lt

Executive Editor

Richard Forster
British Informatics Olympiad, UK, forster@olympiad.org.uk

Technical Editor

Tatjana Golubovskaja
Vilnius University, Lithuania, tatjana.golubovskaja@mii.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
Mile Jovanov, Sts. Cyril and Methodius University, Macedonia,
mile.jovanov@finki.ukim.mk
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.

http://ioinformatics.org/oi_index.shtml

ISSN 1822-7732 (Print)
2335-8955 (Online)

© International Olympiad in Informatics, 2018

Vilnius University, 2018
All rights reserved

Foreword

The International Olympiad in Informatics (IOI) is an annual international informatics competition for individual contestants from over 80 invited countries, accompanied by social and cultural programs as well as a half-day scientific conference for delegation leaders, organisers and guests. The IOI community has an excellent opportunity to communicate during this international event. Many countries have a variety of things to present and discuss. The national olympiads do not exist in isolation, and the papers from the 12th IOI conference show how similar problems arise in different countries and different environments.

The IOI journal is focused on the research and practice of computing professionals who work in the field of teaching informatics to talented secondary and high school students. The journal is closely connected to the scientific conference annually organized during the IOI. The 12th volume has two tracks: the first section of the journal focuses on research, and the second section is devoted to sharing national experiences.

This year IOI is taken place in Tsukuba, Ibaraki, Japan from September 1st to September 8th, 2018. Therefore focused attention is given to informatics education in Japan. T. Kakeshita from Saga university has conducted a “National Survey of Japanese Universities on Computing Education: Analysis of Departments Majored in Computing Discipline”, and H. Manabe, S. Tani, S. Kanemune, Y. Manabe has presented a paper on “Creating original Bebras tasks by high school students”. Y. Nakano and K. Izutsu discuss „The next Course of Study from 2022 and a prospect of information studies education in Japanese senior high schools”. Y. Nakayama, Y. Nakano, Y. Kuno, B. T. Wada, H. Kakuda, M. Hagiya, and K. Kakehi analyse “Current Situation of Teachers of Informatics at High Schools in Japan”.

M.C. Fotaine presents a long study on “Tidal Flow: A Fast and Teachable Maximum Flow Algorithm”.

D. Ginat together with his colleagues H. Galili and N. Lavee discuss a “Algorithmic cognition and pencil-paper tasks”, which underline the aspects of abstraction, heuristics, creativity, and declarative conceptions.

Some of the other papers in this volume deal with teaching programming at primary schools, combinatorial property of sets of boxes in multidimensional Euclidean spaces and theorems in olympiad tasks. A new approach for comparison of countries’ achievements in science olympiads is presented by J. Jovanov, M. Mihove, B. Kostadinov, and E. Stankov. The same authots (except M. Mihove) wrote an article “Platform for analysing and encouraging student activity on contest and e-learning systems”

We understand the need for continuing to share our national experiences – our problems are common problems. In the second part of the volume, M. Anderle from Slovakia, N. Amaroli, G. Audrito, L. Laura from Italy, and several authors from Japan presented their experience. M.S. Tsetvova and V.M. Kiryukhin informed about an international junior school in informatics for IOI training.

Many thanks to the Editorial Board of the IOI journal and also to all those who had assisted with the volume – especially authors and reviewers. A lot of work is required there by starting from writing papers until finishing their final collection for the volume.

In particular, we would like to thank the organisational committee for IOI'2018 in Tsukuba, Ibaraki and the Japanese organisation of this year's IOI for giving us the opportunity to host the IOI conference.

Editors

Computer Science in K-12 Education: The Big Picture

Tim BELL

University of Canterbury, Christchurch, New Zealand
e-mail: tim.bell@canterbury.ac.nz

Abstract. As topics from computer science are increasingly being taught in K-12 schools, it is valuable for those teaching within new curricula to be aware of the purpose of the various components that students are expected to learn. We explore the main purposes of having computer science in curricula in the first place, and then use examples to show how particular topics that might be regarded by some as esoteric can be related to the bigger picture of what is trying to be achieved. The model used is to relate curriculum content to how it affects people, both those who are learning the subject, and those who will be using digital technologies developed by those who have just learned to develop them. This provides a framework to help teachers to motivate themselves, their students, and other stakeholders to engage with new curriculum content.

Keywords: CS education, curriculum, teaching programming.

1. Introduction

Until recently it has been rare for K-12 students to have topics relating to computer science as part of a formal national or state curriculum. However, recently many countries have been introducing topics such as programming and computational thinking (Hubwieser, Giannakos, Berges, *et al.*, 2015; Duncan and Bell, 2015; Heintz *et al.*, 2016). There are several motivations for this, which can loosely be divided into growing students' interest to increase the uptake of the subject by students, and building up knowledge and skills to support students' careers.

As can happen with any curriculum, there is a risk that students see multiple topics spread over a number of years as a disjoint set of independent academic ideas and practical skills that don't necessarily have an obvious purpose. Teachers themselves also need to recognise why a topic is important for their students to understand, and others in the community (including parents and school officials) will also need help to form views on the relevance and importance of new curriculum content.

In this paper we will explore the bigger picture of what curricula should be trying to achieve, and the purpose of including various components in typical curricula, so that all of those involved in it can keep a vision of the broader purpose.

2. The Purpose of K-12 Curricula

As topics relating to computer science enter international curricula, the public perception of the content is often simplified to the term “coding”, or sometimes referred to as “computational thinking” to show that there is more depth to the subject. In reality, most countries are adopting broader curricula that cover issues such as the performance of algorithms, data representation and data structures, software engineering, networks, security, and more (Hubwieser, Giannakos, Berges, *et al.* 2015; Duncan and Bell, 2015; Heintz *et al.*, 2016).

Even if only focussing on programming, the goal should be considerably more than learning to code in some particular language(s). A key goal should be to help students find out if it is something that they might be passionate about, and to give them a vision of what they might do with the wide possibilities that skill in computer science and programming can open up. This is particularly important because stereotypes of who might be good at “coding” can deter those who would enjoy it from even trying (Overdorf and Lang, 2011). Helping students to explore the subject means that it is important that it is presented in a way that they can see the purpose, and appreciate that programming is a skill that supports problem solving and creativity, and can be applied to many areas.

Making programming accessible to those who might not think they are interested involves a delicate balance between supporting students to have early success, while being careful to help them see that it is a skill that will take many of hours of experience to build competence. This relates to Papert’s “low floor” and “high ceiling” (Resnik, 2009). There are wide ranging views on how long it takes to learn programming, and in fact, there are many definitions of what it means to be a competent programmer; in a commercial environment it might mean knowing how to solve clients’ problems using a particular language or software stack, while in a research context or programming competition it could involve having a wide range of skills around implementing sophisticated and novel algorithms. Students who are new to programming need to maintain confidence as they learn, but both under- and over-confidence can lead to them being disheartened and giving up. Peter Norvig (director of research at Google) wrote an article titled “Teach Yourself Programming in Ten Years”,¹ which tries to provide a balance to books that give the impression that programming can be learned very quickly, with titles such as “Teach Yourself Java in 24 Hours”. Of course, the message isn’t that one should plan to spend 10 years learning before becoming a software developer, but that, like any skill, there is always more to learn, and mastery can involve a lot of experience and learning – the more we learn, the more we realise that we don’t know! Having the

¹ <http://norvig.com/21-days.html>

topic in the school curriculum gives students more opportunity to develop skills gradually over time, and also to find out the accompanying skills that are needed to support their programming (such as math for analysing algorithms, communication skills for working with other programmers, and logic for reasoning about program correctness).

The success of helping students discover their passion for computer science can depend on the way it is taught, as can happen with any other subject. This means that having well-prepared teachers with a good understanding of the purpose of the curriculum is key to its success.

For those who have already discovered their passion, there is a need to support them through clubs and competitions that allow them to extend their skill beyond the basics covered in school curricula. In the past, such organisations have provided a key opportunity for students passionate about computing to build relationships with like-minded friends and experience success in a context where they are understood. Such organisations are likely to become *more* important as the subject becomes formalised in K-12, since there will always be those who may become bored with the basic curriculum that is designed for typical students, and need opportunities to stretch themselves. Giving students a vision of what they can do is fundamental to education, and clubs and competitions give students a chance to expand their vision by providing the opportunity to interact with a range of people, including industry professionals, experienced academics, and club/competition alumni.

As well as helping students discover their passion, having computer science as a subject in schools also gives students more time to develop their skills over a period of several years. As with other subjects, knowledge in this area doesn't necessarily have the goal of preparing students for a career in computing, as a general understanding of how digital systems work and what programming is can help students to function in an informed way in an increasingly digital society.

For example, many employees work with spreadsheets as part of their day to day work, yet it is well established that many of the spreadsheets used in practice contain substantive errors (Powell, 2008). This is not surprising when spreadsheet design is seen as having a significant overlap with the skills required to do programming, including proper testing, use of logic in selection (if) commands, and understanding the difference between constants and variables including using effective naming.

Broader topics such as Artificial Intelligence give students a chance to understand a little of the mechanisms behind these apparently mysterious technologies, particularly in the light of the ethical and moral decisions that society is increasingly likely to have to make as the capabilities of such systems grow.

In addition to benefiting students by helping them find out if computing is an area that they have a passion for, new curricula are also intended to benefit broader society, including industry and national interests, by addressing talent shortages and encouraging creativity and entrepreneurship, as well as increasing national digital capabilities that are increasingly needed to maintain security and financial stability for a country.

Whether the motivation is to benefit students or society, the reason for offering a computing curriculum is ultimately to benefit people, and ideally society is better off for having increased capability in this area. Of course, digital systems don't always

have a positive benefit, and students should also be made aware of the ethics surrounding the development of new technologies so that they can contribute positively to society.

3. Seeing How Curriculum Elements Fit Together

We now consider how to see the bigger picture of the curriculum when classes are necessarily focussed on specific topics. A range of curricula are being developed around the world in the area of computing; some refer to the topics as computational thinking, digital technologies, and computer science. All such curricula that we are aware of include programming as a key subject, and most include a range of related topics. For example, an analysis of curricula by Duncan and Bell (2015) classified other common components as:

- Algorithms (designing programs and/or understanding computational complexity).
- Data representation (representation of various data types, and encoding through encryption, compression and error corrections).
- Devices and infrastructure (including device architecture, networks and cloud computing).
- Digital applications (such as simulation, modelling, and creating digital content).
- Humans and computers (including cybersafety, ethics, careers, and human-computer interaction).

A related project has identified ten “big ideas” in computer science that were collated by seeking input from a range of computer science education researchers and professionals from around the world (Bell *et al.*, 2018). The central big idea identified was that “digital systems are designed by humans to serve human needs”; other ideas covered topics such as data representation, algorithms, complexity, computability, virtual representations, time dependent operations, and communication protocols. These aren’t intended to be exhaustive list of what there is to know, but were considered by professionals to be key ideas that students would benefit from being aware of.

With such a diverse range of topics, it’s easy to miss seeing the forest for the trees – learning can become a lot of small topics that can lose connection to the big picture. In the following sections we consider three topics, and relate them to the bigger picture to illustrate these important connections. We build on the central ideas of the discipline being human-centric to identify the value or purpose of the topics.

4. Example: Programming

Computer programming is often characterised as “coding”, but it is a very broad range of skills, and developing software can be broken into elements such as analysis, design, coding, testing, and debugging. Each of these is a discipline in itself; analysis of a situ-

ation for which software is to be developed requires very good communication with the people requiring the program; design involves creativity; testing requires rigour; and debugging requires persistence. Characterising programming as simply “coding” can hide these important human-centric qualities.

Likewise, seeing computer science education as teaching a particular language also risks missing the point. Instead of, say, “teaching Scratch²,” we should think of it as “teaching programming, using Scratch”; or instead of “teaching C++”, we might be “teaching OO programming using C++”. This helps to focus on the broader skills such as testing and debugging, rather than seeing a programming language as a piece of software with some set of features to be learned.

Another important attitude to develop is that we don’t write computer programs for computers; we write them for people! This puts the focus on getting the interface right; this isn’t just about GUI and mobile interfaces, but even for a simple text or dialogue box interface it could involve making sure that it is fast enough to have a good response time for a user – 0.1 seconds is ideal as it will appear instantaneous, otherwise 1 second is a good target to keep interactions at a conversational pace (Johnson, 2013). The interface design could also take into consideration how input and output are presented, such as having simple messages to the user that are unambiguous.

In addition to writing programs for the end user, we also write them for the next programmer who may need to use or modify it (and the next programmer might be the original author in a year’s time, or even the next day). For example, Fig. 1 shows a short Python function; the reader is encouraged to work out what it does before reading on.

Fig. 2 shows the same function with a meaningful name and comment (which makes the purpose of the function explicit), as well as a variable name that has a role in explaining the algorithm. In fact, the name “highest_so_far” could be the basis of an invariant for a proof of correctness for the algorithm, and helps the reader to see what the intended invariant is. Thus by using carefully chosen language in variable and function names, the programmer can help to organise their own thoughts as well as capture the logical intent of the program. Using clearer language also makes it easier to see a slight inefficiency – this implementation makes n key comparisons when only $n - 1$ are required.

Counter to common stereotypes, being an effective programmer requires good social and communication skills, whether the relationship is with the user, or with other

```
def hs(s):
    xxy = s[0]
    for xxx in s:
        if xxx > xxy:
            xxy = xxx
    return xxy
```

Fig. 1. A Python function.

² Scratch is a popular language for teaching programming to beginners.

```
def highest_score(scores):
    """takes a list of scores and returns the highest one"""
    highest_so_far = scores[0]
    for score in scores:
        if score > highest_so_far:
            highest_so_far = score
    return highest_so_far
```

Fig. 2. The Python function of Fig. 1 using better style.

programmers. Blackwell points out that “many skills of a professional programmer are related to social context rather than the technical one” (Blackwell, 2002). The point of learning programming is to develop something to help people, and the rules and conventions that we teach (such as commenting code) are there to support the people who may need to work with the program in the future.

5. Example: Formal Languages

Formal languages appear in some school curricula, and can sometimes be seen as a theoretical topic that is difficult to understand and doesn’t have practical relevance. Here we consider one of the basic ideas of formal languages that often appear in introductory material: regular languages, which can be characterised by regular expressions and parsed by finite state automata (FSA).

Understanding regular languages through a FSA has been introduced at primary school level through a CS Unplugged activity called “Treasure Hunt”,³ where students create a map of ship routes between islands to try to find their way to Treasure Island. The CS Unplugged activity involves several students, and running around a playground; a similar activity that can be used by individuals is provided in the Computer Science Field Guide,⁴ where a student explores an interactive map of commuter train stations.

Follow-up activities enable students to explore conventional FSAs and the regular languages that they represent. But the bigger picture can be found by thinking about how these affect people. We find regular languages being used for checking user input (such as the format of an email address or URL), and most programming languages offer support for regular expressions. They are also fundamental to implementing programming languages by providing lexical analysis based on a human view of the rules (through a regular expression) rather than a computer-centric version (which is the equivalent FSA).

Regular languages are also a gateway to the Chomsky hierarchy, which in turn addresses philosophical questions of what computing is and what can and cannot be computed (the Turing Machine is based on a FSA, and is regarded as a useful abstract model

³ <https://classic.csunplugged.org/finite-state-automata/>

⁴ <http://csfieldguide.org.nz/en/chapters/formal-languages.html#finite-state-automata>

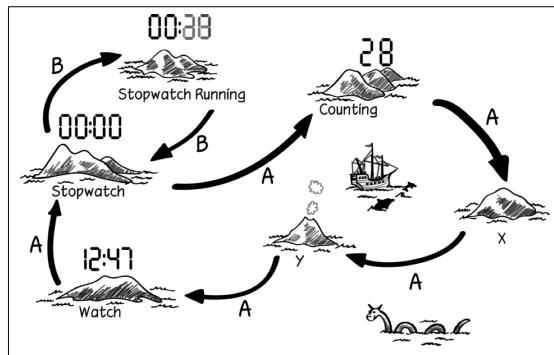


Fig. 3. A “treasure map” (FSA) that represents a digital watch interface.

of computation). This raises very human questions about what these devices are capable of and how the power and limitations of computation may affect us in the future.

An FSA can also be used to model interfaces; for example, Fig. 3 shows a whimsical “map” of the interface for a simple digital watch with two buttons (A and B) that change the state of the watch. This helps us to see an interface as a machine, rather than just a static layout of GUI elements, and can also identify unsafe transitions in more critical interfaces such as medical devices (Thimbleby, 2009). The impact on humans here ranges from mild frustration with a poor interface, to life-and-death situations.

6. Example: Error Detection and Correction

The final example we give is exploring algorithms for error detection and correction. An initial experience with error detection can be given in the classroom through the CS Unplugged Magic Parity Trick.⁵ This involves a grid of 6 by 6 or more cards that can be flipped to show black on one side and white on the other (each card represents one bit). The “magic” trick uses parity forward error correction to determine which card has been flipped while the presenter was looking away. A related “trick” is to work out the check digit at the end of a product code; the presenter can pretend to “mind read” what it is, but of course it can be calculated from the other digits⁶.

Error correction can be demonstrated by “damaging” a QR code. For example, the QR code in Fig. 4 contains the text of the first official Morse code message sent in 1844. To demonstrate error correction at work, the demonstrator can change some of the white “bits” to black with a pen. The second QR code in Fig. 4 has had quite a few of the bits changed, and yet can still be scanned correctly. The third one is probably too damaged to scan; finding the amount of “damage” that can be done while still having the code

⁵ <https://classic.csunplugged.org/error-detection/>

⁶ <https://csunplugged.org/en/topics/error-detection-and-correction/unit-plan/product-code-check-digits/>



Fig. 4. A QR code with various amounts of “damage”.

readable is an interesting exercise for students. The important point is that changing the binary representation doesn’t change the message; without error correction, changing just one bit would change a character in the message.

The bigger picture of error control is again determined by thinking about how it affects people. Binary digits are stored and transmitted in very vulnerable situations, and small physical issues can change single bits easily. If one bit in a file was changed without the user knowing, then the data would be incorrect; a one-bit error could have serious consequences for the people concerned, whether it is a financial amount, exam result or medical data. We take it for granted that data will remain intact, or at worst, the computer will refuse to read a whole file even if only part of it has an error. Another important aspect is that the cost of error correction is considerably less than making full backups; if error correction techniques weren’t used, multiple backups would be needed to have any certainty around the accuracy of data, and we would be constantly comparing copies of the data to check its integrity. The human costs here are the financial cost of extra storage and the delays needed for checking, or if errors are ignored, the impact on the user would be that they are working with incorrect data.

7. Conclusion

A central idea of the “big picture” of computer science is that it is about *people*: “digital systems are designed by humans to serve human needs” (Bell *et al.*, 2018). Programs are written for people, both the users of the program, and the next person who must maintain it. While there are some circumstances where an individual might be solely responsible for an entire program, such as programming exercises and challenges, or encapsulated components of a larger system, the reality is that programming generally leads to an outcome that will affect people. The same is true of all other topics in computer science, whether it is finding an algorithm with better time complexity (so that people can have their problems solved faster and with less demand on computing resources), or implementing checking so that data collection, storage and transmission happen accurately. And at an even broader level, the reason that we teach these subjects is to help students develop a vision for their own future, and to empower them to help others using skills

acquired that enable them to work effectively with digital systems, whether evaluating them critically or creating new systems.

In a subject where the focus is inevitably digital devices, it is good to stand back regularly and see the big picture. For any topic being taught, we want to be able to answer the question “How will someone potentially be better off as a result of knowing this?” with something other than just getting good exam grades or a qualification; the topics in curricula have been chosen because they are tools to benefit people, so the challenge is to be aware of what those benefits are.

References

- Bell, T., Tymann, P., Yehudai, A. (2018). The Big Ideas in Computer Science for K-12 Curricula. *Bulletin of EATCS*, 1(124). Retrieved from <http://bulletin.eatcs.org/index.php/beatcs/article/view/521>
- Blackwell, A. (2002). What is programming? In: *14th workshop of the Psychology of Programming Interest Group*. 204–218.
- Duncan, C., Bell, T. (2015). A Pilot Computer Science and Programming Course for Primary School students. In: *WIPSCE*, 39–48. <http://doi.org/10.1145/2818314.2818328>
- Heintz, F., Mannila, L., Färnvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. In: *Proceedings Frontiers in Education Conference (FIE)*. 1–9. <http://doi.org/10.1109/FIE.2016.7757410>
- Johnson, J. (2013). *Designing with the mind in mind: simple guide to understanding user interface design guidelines*. Elsevier.
- Overdorf, R., Lang, M. (2011). Reaching out to aid in retention. In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education – SIGCSE '11*. 583. <http://doi.org/10.1145/1953163.1953325>
- Powell, S. G., Baker, K. R., Lawson, B. (2008). A critical review of the literature on spreadsheet errors. *Decision Support Systems*, 46, 128–138. <http://doi.org/10.1016/j.dss.2008.06.001>
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Com. of the ACM*, 52(11), 60–67.
- Thimbleby, H. (2009). Contributing to safety and due diligence in safety-critical interactive systems development by generating and analyzing finite state models. In: *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. 221–230.



T. Bell is a professor in the Department of Computer Science and Software Engineering at the University of Canterbury, where he leads the Computer Science Education Research Group. His “Computer Science Unplugged” project is being widely used internationally with the supporting materials (books and videos) having been translated into over 20 languages. Tim has received awards for his work in computing education including the 2018 ACM SIGCSE Outstanding Contribution to Computer Science Education award. He has been actively involved in the design and deployment of the new digital technologies curriculum in New Zealand schools.

How to Start Teaching Programming at Primary School

Michael DOLINSKY, Mariya DOLINSKAYA

*Faculty of Mathematics and Technologies of Programming, Fr. Skorina Gomel State University
Sovetskaya str., 104, Gomel, 246019, Republic of Belarus
e-mail: dolinsky@gsu.by, mkugejko@gsu.by*

Abstract. This article describes the authors' approach to start teaching programming at the primary school, which is based on using distance learning site DL.GSU.BY for sequential leaning seven keywords (program, var, longint, begin, readln, writeln, end) of programming language Pascal. Then these words are using to write the simplest programs that read some numbers, do necessary calculations and write the answer.

Keywords: primary school, programming teaching, distance learning tools.

1. Introduction

Many years the authors prepare scholars of Gomel region for Olympiads in Informatics. All work is doing on the base of distance learning system DL.GSU.BY, created at the Faculty of Mathematics and Technologies of Programming of F. Scorina Gomel State University. Since 2007, training begins with the first grade of primary school. The first stage of training – development of thinking of preschool children and children of primary school age is described in (Dolinsky, 2014). This article presents the authors' approach to learning the first keywords used for Pascal programs: “program, var, longint, begin, readln, writeln, end” and their translation into Russian, as well as their order in the first program. Fig. 1 shows our training goal.

One can see that the exercise was elaborated in the year 2007. Long-term experience of teaching children in programming using this method allowed us to better understand the problems of studying the subject, to improve and develop the learning system.

Chapter 2 presents technology of training. Chapter 3 describes materials for work at the table. Chapter 4 contains remarks on personal learning and teaching in the described process. Finally, chapter 5 contains conclusions.

		учи (-): Учи слова (медленно) Собрать «Все слова» по цвету\3 - "Все слова по цвету" 59126 Кугейко Мария, ПО-22, июль 2007 Назначить авт.				
1		программа program	программа	program	PROGRAM	
2		переменная var	переменная	var	VAR	
3		число longint	число	longint	LONGINT	
4		начало begin	начало	begin	BEGIN	
5		читать readln	читать	readln	READLN	
6		писать writeln	писать	writeln	WRITELN	
7		конец end	конец	end	END	

Fig. 1. Goal of teaching.

2. Technology of Training

Thus, a complete list of children's problems when learning the key words for writing the first Pascal program looks like this:

- To remember the order of the keywords.
- To remember correct spelling of the keywords.
- Learn how to type words on the keyboard.
- To remember translations of the keywords into Russian.

To simplify the memorization of the order of commands, we began to study them using a color background corresponding to the arrangement of colors in the rainbow: red, orange, yellow, green, light blue, blue, purple.

To simplify the memorization of the meaning of the keyword, we select the corresponding image, the meaning of which is explained to the child, if necessary (Fig. 2).

All exercises follow in the order of appearance of the new keyword. I.e. at the beginning there are exercises for "program", then for "var", then for "program – var", then for "longint", then for "program – var – longint" and so on. Special group of the exercise helps memorize letters and their order in keywords. In the process of training, an "unexpected obstacle" appeared – it is customary to type the keywords with lowercase letters but the uppercase letters are written on the keyboard and for many letters their lowercase and uppercase letters are not alike.

Fig. 3 shows an exercise aimed at solving such problems:

To do the exercise one needs to click onto letter for moving, as a result it becomes of red color. Then one needs to click on the button "<<<" to move the letter left or click on the button ">>>" to move the letter right.

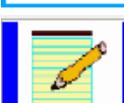
	Program	Bear cub sits at the computer and type a program
	Var	The magician takes out different objects from the tall hat, and they change all the time, a bunny, ribbons ...
	longint	The painted digit “one” is well associated with the word number
	Begin	Alarm clock rings at the beginning of the day
	readIn	The book is opened to read it
	writeln	Pencil writes on paper
	End	The stop sign means the end of the program

Fig. 2. Meaning of the pictures.

<p>Поставьте буквы по образцу:</p> <p>PROGRAM</p> <p>RGPOMRA</p> <p><<< >>></p>	<p>Поставьте буквы по образцу:</p> <p>program</p> <p>rgpomra</p> <p><<< >>></p>	<p>Поставьте буквы по образцу:</p> <p>program</p> <p>RGPOMRA</p> <p><<< >>></p>
---	---	---

Fig. 3. Exercises for permuting letters.

One can see that at the left exercise you need to reorder uppercase letters with uppercase example, at the exercise in the center you need to reorder lowercase letter with lowercase example, and at the right exercise you need to reorder uppercase letters with lowercase example. In the first and second exercises a child may simply use example immediately, but to do the third exercise he needs to remember correspondence of the uppercase and lowercase letters.

Practice showed that for many children it is not enough to do the exercises presented above so we add a set of different auxiliary exercises shown below.

Fig. 4 presents exercise where one also needs to form the keyword “program” by letters permutation, but there is some help: a keyboard is shown on which the letters on the keyboard, which you need to type at the moment, are marked with a yellow color. If the letters are typed correctly, they are green. If the child makes a mistake, the color of the letters becomes red. The letters at input are typed as lowercase letters, showing which lowercase letter correspond pointed uppercase letter.

Fig. 5 presents exercise for memorizing the letters and their order in the keyword PROGRAM. The exercise starts from one letter, then come the second, the third, etc. Fig.5 shows exercise (the eighth exercise in the packet), where one needs to put the first four letters of the keyword PROGRAM in right order. The first letter is given, so the cell for the second letter appears. Note that the wrong letters do not move, so the child must find the right letter and move it to the right place. When the last fourth letter is put in place, the exercise appears with the first 5 letters of the keyword PROGRAM, etc.

The next problem that must be solved to type keywords (after remembering letters and their order) is to remember where the letters are at the keyboard. To solve that problem we create the following exercises, represented at Fig.6 and Fig. 7.

In the exercise in Fig. 6 (left side) one needs to move each letter of the keyword onto its position at the keyboard. Note that if a letter is moved onto its position, the letter jumps there.

In the exercise in Fig. 6 (right side) one needs to move each letter of the keyword from its position at the keyboard onto a cell of the keyword. If a child made an error the letter jumps to its source position on the keyboard.

Additional tips that one can get by clicking onto button “Don’t know” is represented in Fig. 7.

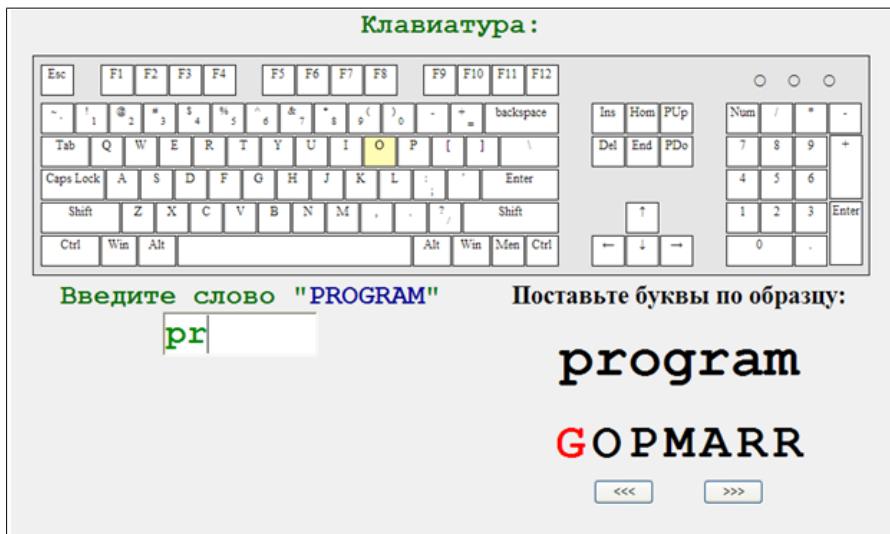


Fig. 4. Type and permute letters of the keyword PROGRAM.

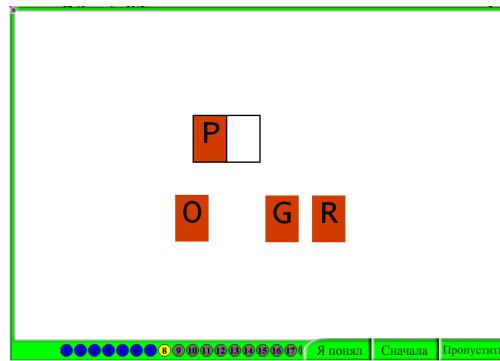


Fig. 5. Compose the word PROGRAM letter by letter.

Fig. 6. Remember letters layout at the keyboard.

Fig. 7. Tips for exercises from Fig. 6.

After the child remembers letters and their order in the keyword, correspondence of lowercase and uppercase letters of the keyword and the letters layout on the keyboard, we can train the keyword typing. To carry out this work the following exercises are offered:

Fig.8 represents exercise for typing keyword with tip. The letter to be typed is indicated in yellow in the picture of the keyboard, the correct printed letters are green, but the wrong printed letter is red.

Fig. 9 represents exercise for typing of absent letters of the keyword. At first the letters are typed in order (first letter, second letter, etc), and then absent letters are in random order.

Fig. 10 shows the exercises that check whether the child remembers the image, the English keyword, the Russian translation, and their mutual correspondence.

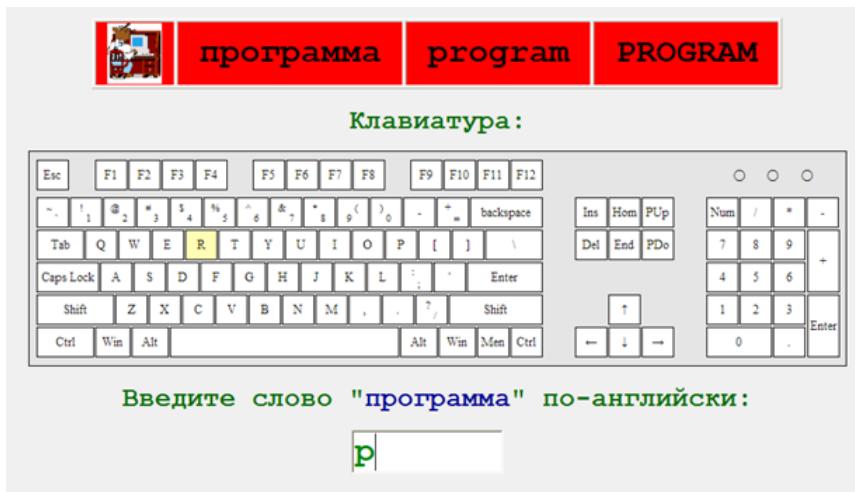


Fig. 8. Typing of the keyword PROGRAM with tip.

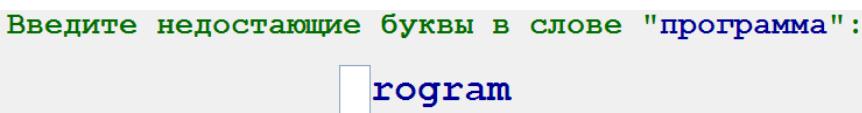


Fig. 9. Typing of absent letters.

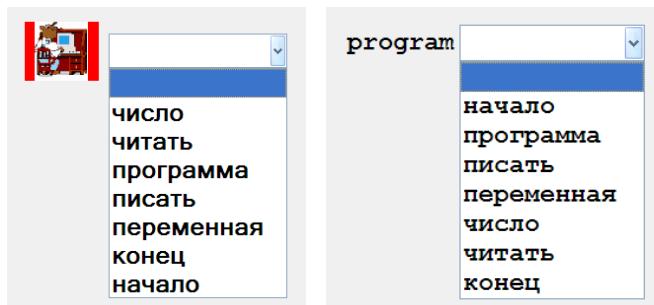


Fig. 10. Correspondence of the Russian word to the image and the English word.

3. Materials for Work at the Table

When we started work in the autumn of 2007, special attention was paid to working at the table. At first we tried to reduce the time for working with computers, so that they would be enough for each child. Secondly, we did not have computer analogues of these exercises.

First of all, this is the workbook “Learn keywords”, presented in Fig. 11. It contains all the keywords you need to remember: pictures, keywords, translations, and a set of exercises for learning and memorizing. For example, join letters of a keyword, point the position of the letter on the keyboard, etc. In addition, such notebooks help to develop skills of correct writing letters (pen on paper) and keywords. The notebook is provided to the child on demand, first of all, for independent work at home. To work in the lessons, as a rule, the materials shown in Fig. 12–17 are used.

Fig. 12, for example, represents some sets of cards that are used for the words to be remembered.

Some exercises:

- Gather the cards in order shown on the Fig. 12.
- Contest “Who will quickly collect a table with keywords from the cards”.
- Find absent card or cards.

Fig. 13 contains keywords table that allows adding the following exercises:

- Gather the cards with example.
- Gather the cards onto keywords table.

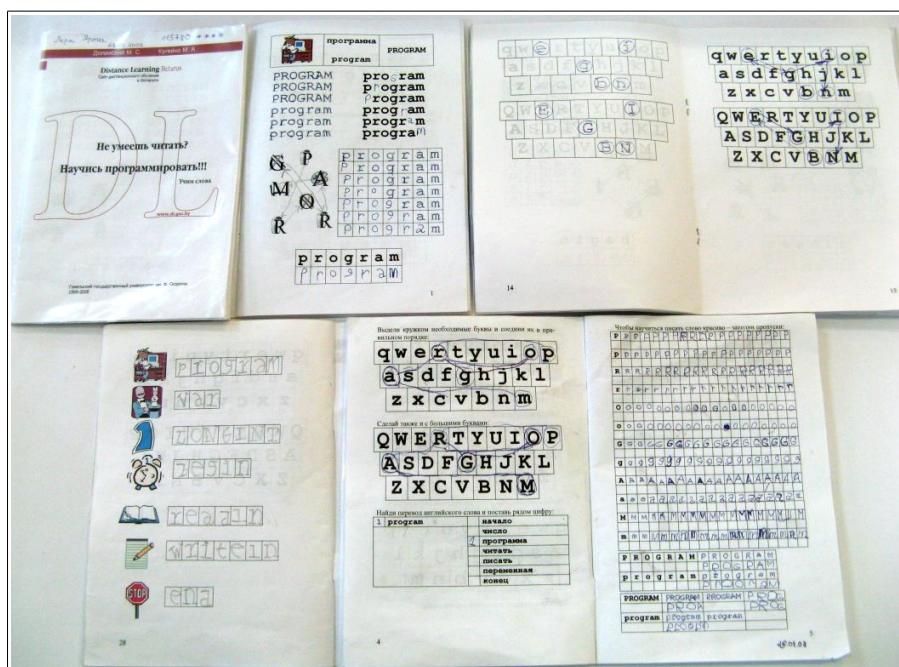


Fig. 11. Notebooks for work at the table.



Fig. 12. Color cards with keywords.



Fig. 13. Color keyword cards with keywords table.

Fig. 14 shows the table of keywords in black and white colors, which makes it difficult to apply the above exercises because of the disappearance of “color support”.

Great attention is given to work with letters of English keywords. Fig. 15 contains cards which allow compiling the keywords from lower- and uppercase letters, with or without samples, for a separate keyword or for all keywords.

Все слова				
1	программа program	программа program	program	PROGRAM
2	переменная var	переменная var	var	VAR
3	число longint	число longint	longint	LONGINT
4	начало begin	начало begin	begin	BEGIN
5	читать readln	читать readln	readln	READLN
6	писать writeln	писать writeln	writeln	WRITELN
7	конец end	конец end	end	END

1	программа program	программа program	program	PROGRAM
2	переменная var	переменная var	var	VAR
3	число longint	число longint	longint	LONGINT
4	начало begin	начало begin	begin	BEGIN
5	читать readln	читать readln	readln	READLN
6	писать writeln	писать writeln	writeln	WRITELN
7	конец end	конец end	end	END

Fig. 14. Black and white cards and sample.



Fig. 15. Box with the keywords letters.



Fig. 16. “Paper” keyboard with letters of PROGRAM.



Fig. 17. “Box” keyboard with keyword PROGRAM.

Fig. 16 represents “paper” keyboard. Children can put letters of given keywords onto it.

Fig. 17 shows the “prototype keyboard”, which is made from glued matchboxes containing letters on the positions of the real keyboard. In this way, we can train memorizing the keyboard layout without a computer.

4. Personal Learning and Teaching

As a result, we created a training system where each child receives exercises that are feasible for him in complexity and at the same time leading (albeit at different speeds) to a common goal – to teach typing key words on the keyboard of the computer in the Pascal programming language: “program, var, longint, begin, readln, writeln, end”, and also remember their order in the program and their translation into Russian. All this together is the best basis for a simple transition to the creation of the first program “reading and writing number”.

Individual training and teaching provide a rich set of exercises at the table, as well as computer-based automatic personal exercises, appointed depending on the results of previous exercises. In addition big set of exercise packets supporting different entry points to learning, including: “Propaedeutics of keywords”, “Learn keywords”, “Auxiliary learning”: “program”, “longint”, “begin”, “readln”, “writeln”. The standard approach is to start with “Keyword Propaedeutics”, then go to “Learn keywords”. If the child has forgotten a keyword, he can repeat it in “Auxiliary Learning” for this keyword.

5. Conclusion

This article presents an authors' approach to starting teaching programming in elementary school, based on the consistent use of seven key words in the Pascal programming language necessary to create simple programs that read some numbers make the necessary calculations and write the answer.

It is important to note that we have many exercises with different levels of difficulty to work both on the computer and at the table. They provide effective individual education and training for children with different levels of training and motivation. The presented system of teaching and learning is scalable and can be used even by teachers and parents who are initially far from programming, as shown by the present practice. Initially, you can conduct joint training. And after that, when the child “gained speed” successfully continue the classes on their own.

References

- Dolinsky M. (2005). *Algorithmization and Programming with TURBO PASCAL: From Simple to Olympiad Problems: Tutorial*. Sankt-Petersburg: “Piter” (In Russian: Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач: Учебное пособие. СПб.: Питер).
- Dolinsky M. (2006). *Solving of Sophisticated Olympiad Programming Problems: Tutorial*. Saint-Petersburg: “Piter” (In Russian: Решение сложных и олимпиадных задач по программированию: Учебное пособие. СПб.: Питер).
- Dolinsky M. (2013). An approach to teach introductory-level computer programming. *Olympiads in Informatics*, 7, 14–22.
- Dolinsky M. (2014). Technology for the development of thinking of preschool children and primary school children. *Olympiads in Informatics*, 8, 63–68.
- Dolinsky M. (2016). Gomel training school for Olympiads in Informatics. *Olympiads in Informatics*, 10, 237–247.
- Dolinsky M. (2017). A new generation distance learning system for programming and Olympiads in Informatics. *Olympiads in Informatics*, 11, 29–39.
- Performance Statistics of Gomel pupils at international and national olympiads in informatics since 1997 up to 2017. (In Russian): <http://d1.gsu.by/olympr/result.asp>



M. Dolinsky is a lecturer in Gomel State University “Fr. Skaryna” 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.



M. Dolinskaya is student in Gomel State University “Fr. Skaryna” from 2005 then graduate student from 2017. Since 2006 she is one of developer of the educational site dl.gsu.by as well as teacher of pupils from first grade. Her current research is in teaching programming from early age.

Tidal Flow: A Fast and Teachable Maximum Flow Algorithm

Matthew C. FONTAINE

Independent Researcher
e-mail: tehqin@gmail.com

Abstract. Among the most interesting problems in competitive programming involve maximum flows. However, efficient algorithms for solving these problems are often difficult for students to understand at an intuitive level. One reason for this difficulty may be a lack of suitable metaphors relating these algorithms to concepts that the students already understand. This paper introduces a novel maximum flow algorithm, Tidal Flow, that is designed to be intuitive to undergraduate and pre-university computer science students.

Keywords: algorithms, maximum flow, flow network, flow augmentation.

1. Introduction

Maximum flows are a well-researched area in optimization theory. The problem was originally formulated by Harris and Ross (1955) and solved by using the well-known augmenting path technique by Ford and Fulkerson (1955). Since this initial discovery, many maximum flow algorithms have been developed (for a survey see: Ahuja *et al.*, 1993; Goldberg and Tarjan, 2014). Most of these algorithms are space efficient. As a result, time complexity becomes the primary basis of comparison between maximum flow algorithms (Goldberg and Tarjan, 2014). The maximum flow formulation opens itself to a wide variety of applications (for examples see: Ahuja *et al.*, 1993). This diversity of applications has increased the popularity of problems involving maximum flows in the competitive programming community. Problems that require the creation of networks with large capacities and a large number of vertices and edges demand the use of faster flow algorithms to solve each problem in a reasonable amount of time. These more complex algorithms can be a burden for students to understand. As proposed by Forišek and Stienová (2013), there are differences among deriving, proving, and teaching algorithms. How easy an algorithm is to teach is a factor in how widely adopted an algorithm will be. Moreover, the theoretically best algorithms with respect to time complexity are not always the fastest when implemented in practice (Ahuja *et al.*, 1997), i.e. in a programming contest setting. These considerations create a need

for a fast flow algorithm that is easy to teach, is easy to implement, and offers competitive in-practice performance compared to other popular flow algorithms. This paper proposes Tidal Flow as an algorithm to satisfy these three objectives.

2. Background

This section reviews the maximum flow problem and introduces the notation that will be used throughout the rest of the paper. It then describes three historically important maximum flow algorithms that introduce necessary concepts for understanding the novel Tidal Flow algorithm.

2.1. Maximum Flow Formulation and Notation.

This paper will use formulation and notation adapted from Goldberg and Tarjan (2014). The input to the maximum flow problem is (G, s, t, cap) , where $G = (V, E)$ is a directed graph with n vertices and m edges. The input marks two special vertices $s, t \in V$. The vertex s is known as a source and the vertex t is known as the sink. The function $\text{cap} : E \rightarrow \mathbb{R}^+$ is some strictly positive capacity function. A maximum flow is some non-negative function $f : E \rightarrow \mathbb{R}^*$ that satisfies two constraints: (1) a capacity constraint $f(e) \leq \text{cap}(e), \forall e \in E$ and (2) a conservation constraint $\sum_{(w,v) \in E} f(w, v) = \sum_{(v,u) \in E} f(v, u), \forall v \in V - \{s, t\}$. The capacity constraint ensures that flow sent down some edge e does not exceed its capacity, while the conservation constraint maintains the flow entering some vertex v equals the amount of flow leaving v . The conservation constraint is maintained for all vertices except the source and sink. The flow value is defined as the amount of flow leaving the source or $\sum_{v \in V} f(s, v)$. A maximum flow is one that maximizes the flow value subject to the conservation and capacity constraints.

2.2. Residual Graphs and Augmenting Paths.

To make it easier to discover maximum flows, it is useful to make the graph more malleable. Residual graphs are a useful tool for this purpose. Consider each edge $(w, v) \in E$. To change the flow along this edge, one could increase the flow by up to $\text{cap}(w, v) - f(w, v)$ or decrease the flow by $f(w, v)$. Decreasing the flow is easier to manage by including a reverse edge (v, w) with $\text{cap}(w, v)$ where $f(v, w) = -f(w, v)$ at all times. Now, decreasing flow $f(w, v)$ can be accomplished by increasing $f(v, w)$. The amount that $f(w, v)$ can be decreased is given by $\text{cap}(v, w) - f(v, w)$. Now consider a new edge set E' that contains all edges of E and all reverse edges of E . The residual graph is $G_r = (V, E')$ with a new function $\text{cap}_r(w, v) = \text{cap}(w, v) - f(w, v)$. Conceptually, cap_r gives a limit on how much each $e \in E'$ can change along that direction.

To modify f and maintain conservation and capacity constraints, an algorithm can discover some path P from source to sink where each edge on that path e has $\text{cap}_r(w, v) > 0$. The algorithm can then augment each edge $e \in P$ by $a = \min_{e \in P}(\text{cap}_r(e))$.

This change can be accomplished through modifying f : $f(w, v) \leftarrow f(w, v) + a$ and $f(v, w) \leftarrow f(v, w) - a$ for each edge $(w, v) \in P$. P is known as an *augmenting path* (Ford and Fulkerson, 1955). Each augmentation maintains the capacity and conservation constraints and gradually increases the flow with each augmentation until a maximum flow is reached. Augmenting paths can be applied between any two vertices $w, v \in V$. The term *global augmenting path* refers to augmenting paths between s and t to distinguish this approach from techniques that make local improvements to flow.

2.3. Dinitz's Algorithm and Level Graphs.

Augmenting along shortest paths was discovered to be more efficient than augmenting along other types of paths (Dinitz, 1970; Edmonds and Karp, 1972). Dinitz's algorithm works by efficiently computing a level graph from s to t and finding paths from s to t through that level graph.

A *level graph* $G_L = (V_L, E_L)$ is a graph where V_L contains all vertices $v \in V$ on all shortest paths from s to t in G (see Fig. 1). Let $d : V \rightarrow \mathbb{Z}^*$ denote the number of edges on a path from s to $v \in V$ with minimum number of edges. E_L contains all edges in $(w, v) \in E$ where $\text{cap}_r(w, v) > 0$ and $d(w) + 1 = d(v)$.

Dinitz's algorithm computes a level graph using a breadth first search (BFS). The algorithm then attempts to saturate enough edges in G_L to prevent any augmenting path from s to t . An edge $e \in E$ is *saturated* if $\text{cap}_r(e) = 0$. Such a flow is called a *blocking flow* of level graph G_L .

Shimon Even revised and popularized Dinitz's algorithm, creating the well-known version (Dinitz, 2006). (When popularizing the algorithm, Even spelled Dinitz as Dinic and changed the pronunciation (Dinitz, 2006).) In Even's version, a blocking flow is computed through a modified depth first search (DFS). The DFS finds a maximal set of shortest augmenting paths, which is sufficient to block G_L . The modified DFS produces these augmenting paths in $O(nm)$ running time. The BFS and DFS procedures repeat until no augmenting path exists from source to sink. Each level graph blocked increases the length of the shortest path in the next discovered level graph, resulting in an $O(n^2m)$ total running time. The original algorithm also computes a blocking flow of this level graph in $O(nm)$, but is both more complicated conceptually and harder to implement.

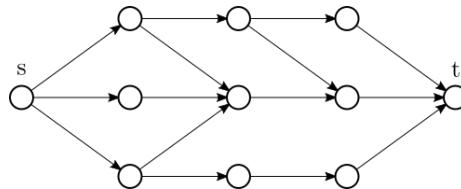


Fig. 1. A level graph.

2.4. Karzanov's Algorithm and Preflows.

Karzanov formalized the concept of blocking flows and additionally introduced the concept of preflows (Karzanov, 1974). In Karzanov's algorithm, preflows are used to in place of global augmenting paths to block the level graph. A *preflow* is similar to a flow f , but the conservation constraint is removed, meaning more flow can go into a vertex than is leaving it. To help keep track of the amount of extra flow at some vertex a function $ex : V \rightarrow \mathbb{R}^*$ is used, where $ex(v) = \sum_{(w,v) \in E} f(w,v) - \sum_{(v,u) \in E} f(v,u)$.

Similar to Dinitz's algorithm, Karzanov's algorithm repeatedly computes level graphs and blocks them. For each level graph, the algorithm sends preflows through the level graph and gradually restores the conservation constraint, converting the blocking preflow into a blocking flow. The blocking flows are discovered in $O(n^2)$ running time, resulting in a $O(n^3)$ running time for the algorithm.

3. Tidal Flow

This section introduces the Tidal Flow algorithm by first explaining its relationship to Dinitz's and Karzanov's algorithms, then introducing an ocean tide metaphor that makes the algorithm easier to understand at a high level, and then finally explaining the technical details and formalization of the algorithm.

3.1. Blocking Flows.

Like Dinitz's algorithm and Karzanov's algorithm, Tidal Flow computes blocking flows on a level graph. Like Dinitz's algorithm, Tidal Flow's goal is to produce blocking flows by discovering global augmenting paths. However, instead of computing the blocking flow in $O(nm)$, the Tidal Flow attempts to compute a blocking flow in $O(m)$. Unlike Dinitz's algorithm, Tidal Flow makes no guarantee to discover a blocking flow in the level graph in one pass. To simplify the blocking procedure, the edges of the level graph are stored as a list in BFS order (Fig. 2).

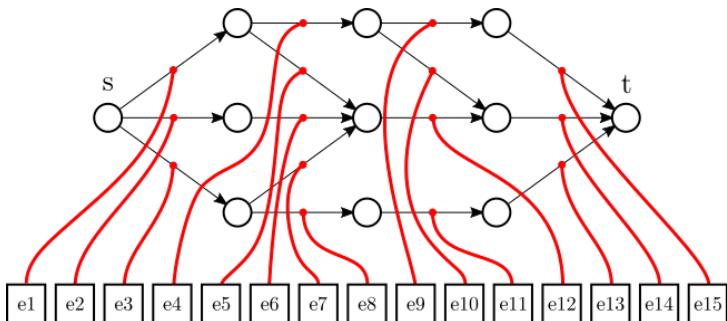


Fig. 2. A level graph stored as a list of edges.

3.2. Tide Metaphor.

Metaphors can be a powerful tool in teaching new algorithms. Forišek and Stienová give the following definition:

A (conceptual) metaphor is a cognitive process that occurs when a subject seeks understanding of one idea (the target domain) in terms of a different, already known idea (the source domain). The subject creates a conceptual mapping between the properties of the source and the target, thereby gaining new understanding about the target. (Forišek and Stienová, 2013, adapted from Lakoff and Johnson, 2003)

Tidal Flow uses a conceptual metaphor based on oceanic tide cycles to help explain its level graph blocking procedure.

3.3. Discovering Blocking Flows as Tides.

The goal of Tidal Flow is to produce a blocking flow on a level graph. Tidal Flow does that through a procedure called *tide cycle*.

Tide cycle has three phases: high tide, low tide, and erosion.

- (1) **High tide:** Produce an upper bound on the amount of flow that can reach each vertex in the level graph by passing from source to sink.
- (2) **Low tide:** Reduce the amount of flow that can reach each vertex to a feasible amount by passing from sink to source.
- (3) **Erosion:** Change the flow on each edge used and update residual flow.

In terms of the metaphor, vertices are *tide pools* that temporarily collect flow during each phase of tide cycle. Tide pools store an upper bound on the amount of flow that can reach each vertex during high tide when flow passes from source to sink. During low tide, flow is pushed back from sink to source. Not all flow will make it back to the source and some tide pools retain excess flow (similar to how tide pools in nature exist as separate bodies of water during low tide). Because the level graph is stored as a list of edges, each phase can be implemented as a loop through the list of edges.

Consider the example level graph in Fig. 3. During the high tide phase, a function $h : V \rightarrow \mathbb{R}^*$ is computed storing an upper bound on the amount of flow that can reach each vertex. In terms of the metaphor, $h(v)$ is the amount of flow stored in

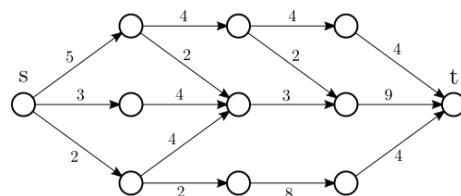


Fig. 3. Example graph with edge capacities.

each tide pool at high tide. $h(v)$ can be any upper bound on flow that can reach vertex v through a set of augmenting paths that collectively maintain the capacity constraint. One could consider h to be a heuristic guessing function for Tidal Flow, similar to the heuristic function in the A* shortest path algorithm (Lerner *et al.*, 2009). For simplicity, $h(v)$ is computed as the sum of $h(w)$ for each edge (w, v) that enters v bounded by that edge's capacity (Equation 1, Fig. 4). Flow is promised to each edge $(w, v) \in E$ of the level graph $p(w, v) = \min(\text{cap}(w, v) - f(w, v), h(w))$.

$$h(v) = \sum_{w \in G(v)} \min(\text{cap}(w, v) - f(w, v), h(w)) \quad (1)$$

During the low tide phase, flow is pushed from sink to source backwards through the level graph using h and p as guides for how much flow to push to each vertex. A new function $l : V \rightarrow \mathbb{R}^*$ maintains the amount of flow in tide pool v . $l(t)$ is initialized with $h(t)$. When an edge (w, v) is evaluated on the way back to the source, flow is drained from $l(v)$ and transferred to $l(w)$ and the promised flow $p(w, v)$ is updated by this transferred flow (Fig. 5).

In the erosion phase, the promised flow is committed to the network. In the example, 11 units of flow were promised from source to sink, but only 9 units of flow are committed. In the example, Tidal Flow manages to find a blocking flow in one pass of tide cycle.

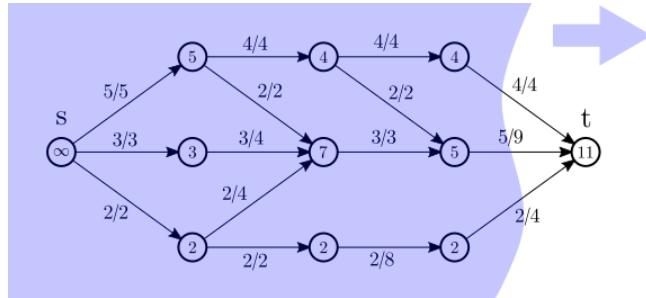


Fig. 4. High tide calculates $h(v)$ for each vertex v . Edges store promised flow $p(e)/\text{cap}_r(e)$.

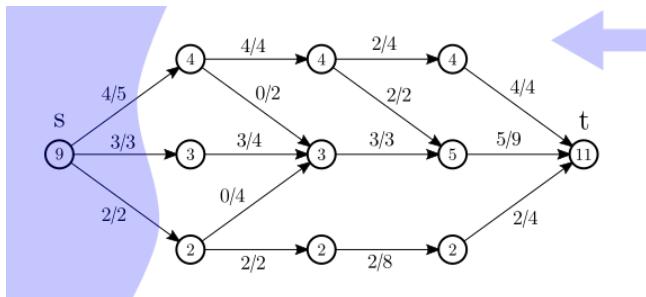


Fig. 5. Low tide pushes flow from t to s . Here each vertex contains $l(v)$ at its highest point during low tide.

3.4. Relationship to Preflows.

The low tide phase of tide cycle is a type of preflow. Excesses are placed at the sink and as much flow as possible is pushed between nodes. The preflow values never augment the network, so Tidal Flow always maintains the conservation constraint. Instead, preflows are used to discover a set of potentially overlapping augmenting paths that collectively maintain the capacity constraint and augment across those paths in parallel. In this sense Tidal Flow is both a preflow algorithm and an augmenting path algorithm.

Algorithm 1: Attempt to compute a blocking flow

TideCycle (E)

```

input : A list  $E$  of level graph edges in BFS order.
output: The amount of flow sent through the level graph.
result :  $f$  is modified by found augmenting paths.
 $h(v) = 0, \forall v \in V$ ;
 $h(source) \leftarrow \infty$ ;
foreach edge  $e_i(w, v) \in E$  do
     $p(e_i) \leftarrow \min(cap(e_i) - f(e_i), h(w))$ ;
     $h(v) \leftarrow h(v) + p(e_i)$ ;
end
if  $h(sink) = 0$  then
    return 0;
end
 $l(v) = 0, \forall v \in V$ ;
 $l(sink) \leftarrow h(sink)$ ;
foreach edge  $e_i(w, v) \in E$  in reverse order do
     $p(e_i) \leftarrow \min(p(e_i), h(w) - l(w), l(v))$ ;
     $l(v) \leftarrow l(v) - p(e_i)$ ;
     $l(w) \leftarrow l(w) + p(e_i)$ ;
end
 $h(v) = 0, \forall v \in V$ ;
 $h(source) \leftarrow l(source)$ ;
foreach edge  $e_i(w, v) \in E$  do
     $p(e_i) \leftarrow \min(p(e_i), h(w))$ ;
     $h(w) \leftarrow h(w) - p(e_i)$ ;
     $h(v) \leftarrow h(v) + p(e_i)$ ;
     $f(e_i) \leftarrow f(e_i) + p(e_i)$ ;
     $f(rev(e_i)) \leftarrow f(rev(e_i)) - p(e_i)$ ;
end
return  $h(sink)$ ;
```

4. Evaluating the Performance of Tidal Flow

4.1. Correctness.

Flow is only modified through augmenting paths, which maintain the capacity and conservation constraints. During each tide cycle, the amount of flow increases until no augmenting paths exist. Using the usual arguments involving augmenting paths, Tidal Flow will terminate with a maximum flow.

4.2. Theoretical Performance of Tidal Flow.

Edmonds and Karp (1972) introduced an argument for bounding the running time of the shortest augmenting path method of finding maximum flows. The tide cycle procedure in Tidal Flow will always fully saturate at least one edge on a shortest augmenting path. This gives an upper bound on the time complexity of Tidal Flow to be at most $O(nm^2)$. This bound may not be tight. Tide cycle will regularly find several augmenting paths.

4.3. Difficulty of Bounding Tide Cycles.

Determining the upper bound on the number of tide cycles required to produce a blocking flow is difficult. During high tide, a network can trick the heuristic function h into promising much more flow than is feasible to realize during low tide by creating a lens (Fig. 6) somewhere in the network. A *lens* is a dense level subgraph with two node levels where edges double in capacity each level.

Lenses require $\approx \log(k)$ nodes to magnify the flow through the network by k . Using lenses, it is possible to construct level graphs requiring $O(n/\log(n))$ tide cycles to block. Fig. 7 provides one such construction. Creating level graphs requiring more tide cycle operations is not obvious.

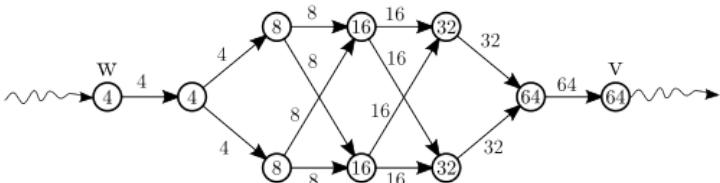


Fig. 6. A lens. Only 4 units of flow can pass from w to v , yet 64 units are promised.

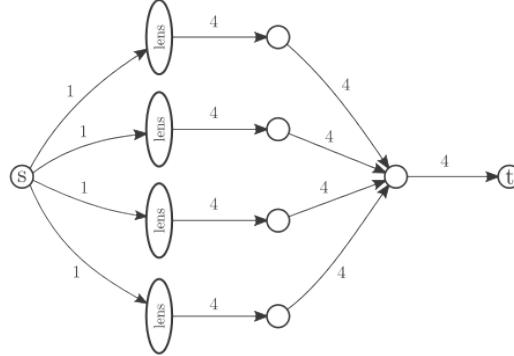


Fig. 7. A level graph that takes 4 tide cycles to block. The graph can be generalized to a graph with n vertices requiring $O(n/\log(n))$ tide cycles.

4.4. Evaluating Practical Performance.

Many of the theoretically fastest maximum flow algorithms are much slower in practice (Goldberg and Tarjan, 2014). The fastest theoretical algorithm for maximum flows is due to Orlin (2013) and achieves a $O(nm)$ running time. The overhead of the algorithm and rarity of the worst cases causes the best theoretical approach to be defeated in practice by slower algorithms (Boykov and Kolmogorov, 2004; Goldberg *et al.*, 2011). Though many flow algorithms have large upper bounds on running time, they rarely achieve this behavior in the average case.

To evaluate Tidal Flow, the algorithm was benchmarked against several maximum flow algorithms (Table 1) that are known to do well in practice (Ahuja *et al.*, 1997)

Table 1
Flow algorithms

Algorithm	Running Time	Notes
Edmonds-Karp (Edmonds and Karp, 1972)	$O(nm^2)$	Shortest augmenting path
Dinitz (Dinitz, 1970)	$O(n^2m)$	Even's version with optimizations suggested in Dinitz (2006)
Preflow-Push (Goldberg and Tarjan, 1988)	$O(n^3)$	Goldberg and Tarjan's preflow-push algorithm with the highest-label selection rule. A simple, but inefficient, selection implementation process yields the slower runtime
Preflow-Push (Gap) (Goldberg and Tarjan, 1988)	$O(n^2 \sqrt{m})$	Goldberg and Tarjan's push relabel method with the highest-label selection rule. Implemented with $O(1)$ selection and the gap relabeling heuristic suggested in Cherkassy and Goldberg (1995)
Improved-SAP (Orlin and Ahuja, 1987)	$O(n^2m)$	Orlin's improved shortest augmenting path method

and are popular in competitive programming. Each flow algorithm was run against a test suite of randomly generated networks that resemble classes of graphs common in competitive programming network flow problems. The algorithms were implemented in Java and each experiment measured CPU time using the StopwatchCPU class from Sedgewick and Wayne (2011).

Three forms of graphs were tested: bipartite matching networks, grid networks, and level graphs (Fig. 8). Bipartite matching networks were split into four different graph classes, resulting in six total graph classes. Each graph class was evaluated at 10 different sizes (described below). For each size of graph in each graph class, 20 random graphs were generated. Each algorithm was run against each test for a maximum of 20 seconds.

- (1) **Dense-highcap-bpm:** A fully connected bipartite matching graph. Capacities for internal edges were selected uniformly at random from [1, 1000]. Capacities for edges (s, v) and (v, t) were selected from $[1, c(v)]$ where $c(v) = \sum_{(v,w) \in E} cap(v, w)$. Graph size n represents the number of internal vertices. A range of sizes $n = [200, 2000]$ were selected in increments of 200.
- (2) **Sparse-highcap-bpm:** A bipartite matching graph. Each vertex v on the source side of the bipartite graph was connected with \sqrt{n} random neighbors on the sink side. Capacities for internal edges were selected uniformly at random from [1, 10000]. Capacities for edges (s, v) and (v, t) were selected from $[1, c(v)]$ where $c(v) = 1 + \sum_{(v,w) \in E} cap(v, w)$. Graph size n represents the number of internal vertices. A range of sizes $n = [1000, 10000]$ were selected in increments of 1000.
- (3) **Dense-unit-bpm:** A bipartite matching graph where all edge capacities are unit capacities. Each vertex v on the source side of the bipartite graph was connected with 10 random neighbors on the sink side. Graph size n represents the number

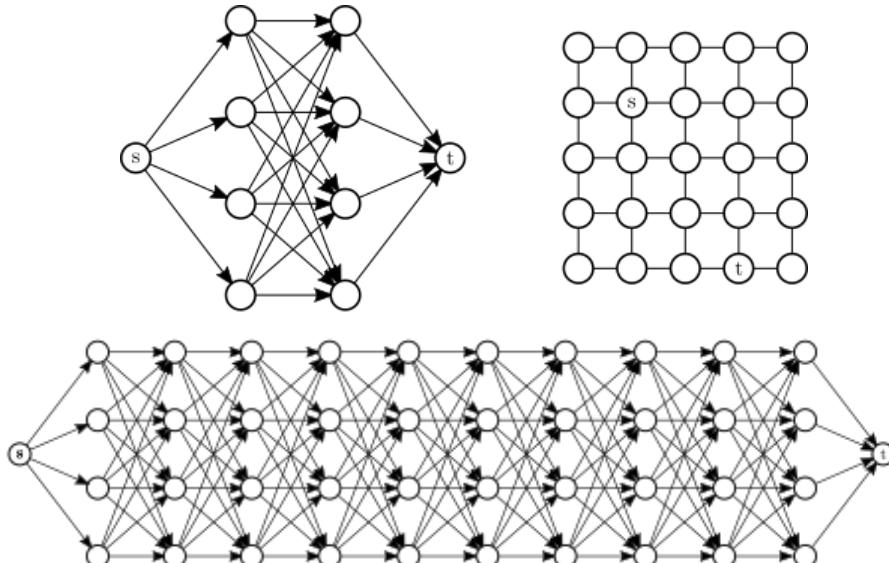


Fig. 8. Three forms of graph classes.

of internal vertices. A range of sizes $n = [28000, 10^5]$ were selected in increments of 8000.

- (4) **Sparse-unit-bpm:** A bipartite matching graph where all edge capacities are unit capacities. Each vertex v on the source side of the bipartite graph was connected with 10 random neighbors on the sink side. Graph size n represents the number of internal nodes. A range of sizes $n = [28000, 10^5]$ were selected in increments of 8000.
- (5) **Grid:** An $n \times n$ grid network where each neighbor in the four cardinal directions is connected. The source and sink were selected uniformly at random among grid vertices. Edge capacities were selected uniformly at random from $[1, 10^8]$. Graph size n represents an $n \times n$ grid of vertices. A range of sizes $n = [275, 500]$ were selected in increments of 25.
- (6) **Level-10:** A level graph with 10 levels where level i is fully connected to level $i + 1$. Capacities for internal edges were chosen uniformly at random from $[1, 1000]$. Capacities for edges (s, v) and (v, t) were selected from $[1, c(v)]$ where $c(v) = \sum_{(v,w) \in E} cap(v, w)$. Graph size n represents a $10 \times n$ level graph. A range of sizes $n = [140, 500]$ were selected in increments of 40.

5. Results

Fig. 9–Fig. 14 compare Tidal Flow against the flow algorithms from Table 1. Flow algorithms that didn't complete a majority of the tests were removed from the figures to make it easier to directly compare Tidal Flow against more competitive algorithms.

Algorithm performance on dense-highcap-bpm (Fig. 9). On this graph class Edmonds-Karp only completes size 200 graphs before timing out on all remaining test sizes. Preflow-Push (Gap) manages to complete up to 1200 size graphs but times out at 1400. The variance of Preflow-Push (Gap) is much higher than other algorithms. Preflow-Push completes graphs up to size 1400, but also has a high variance. ISAP completes graphs up to size 1400. Dinitz manages to complete all tests but performs worse

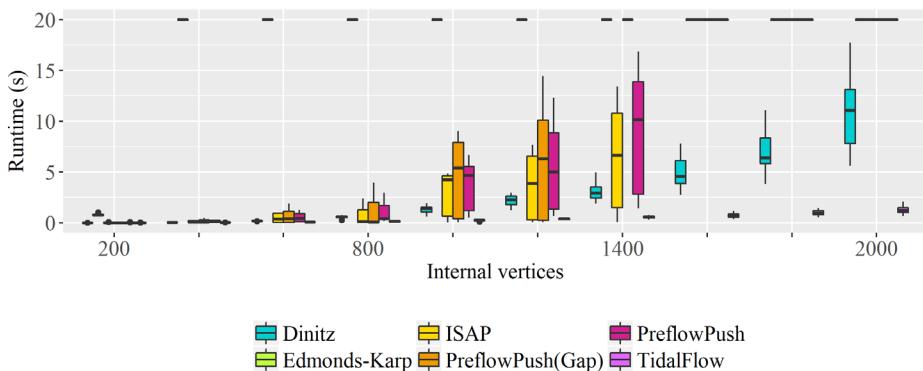


Fig. 9. Algorithm performance on dense-highcap-bpm.

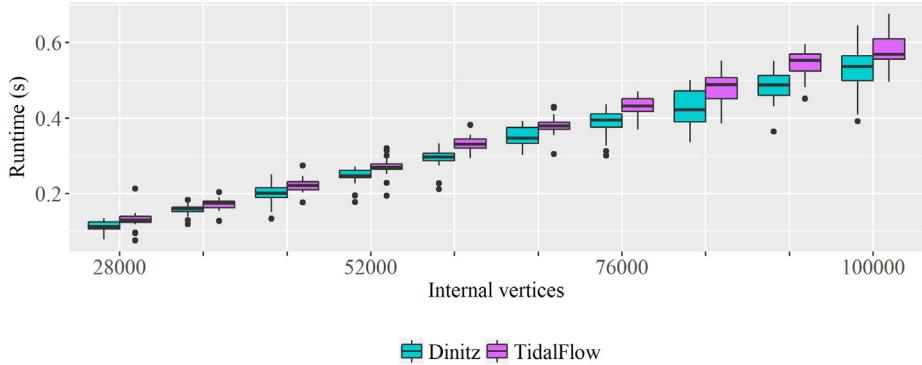


Fig. 10. Algorithm performance on sparse-highcap-bpm.

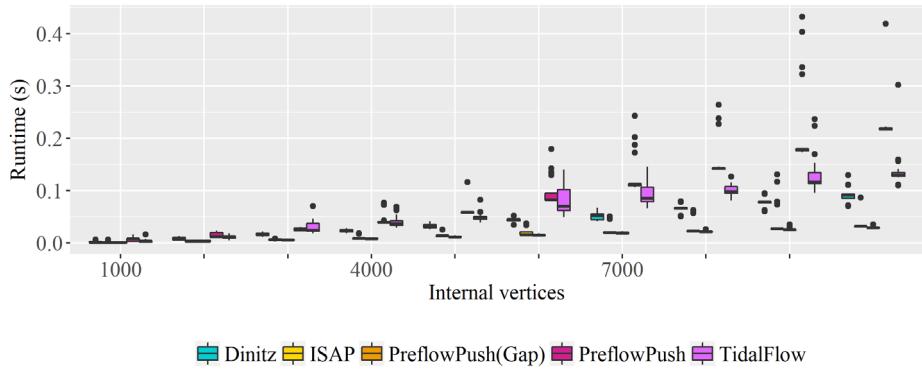


Fig. 11. Algorithm performance on dense-unit-bpm.

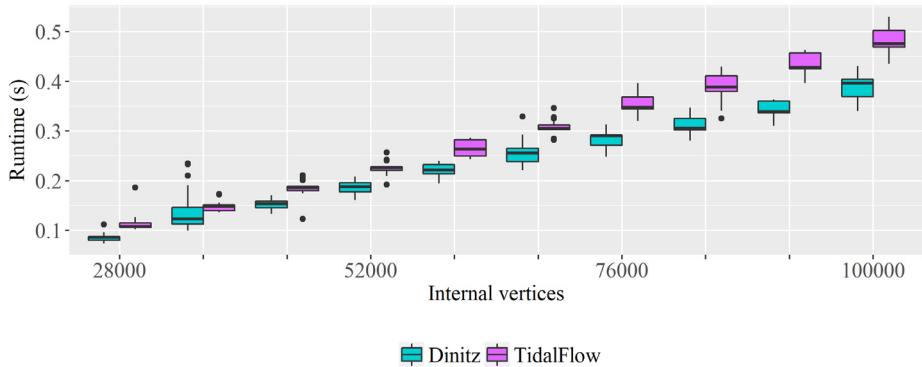


Fig. 12. Algorithm performance on sparse-unit-bpm.

than Tidal Flow. Tidal Flow completes all tests and has much lower variance than any of the other included algorithms.

Algorithm performance on sparse-highcap-bpm (Fig. 10). On this graph only Tidal Flow and Dinitz were able to complete all tests in 20 seconds. Preflow-Push(Gap) was the only other algorithm able to complete a test and only completed the smallest test size in under 12 seconds before timing out. Only Dinitz and Tidal Flow are included in this graph to allow a closer comparison of the two algorithms. Dinitz performs slightly better than on this graph class, but Tidal Flow is comparable in performance. All tests run under 0.7 seconds.

Algorithm performance on dense-unit-bpm (Fig. 11). Dinitz completes all tests in under 0.2 seconds. Edmonds-Karp completes half of the test suite before timing out. ISAP runs all tests in under 0.04 seconds. PreflowPush (Gap) runs all tests in under 0.03 seconds. Preflow-Push runs all tests in under 0.3 seconds. Tidal Flow is comparable to Dinitz, solving all tests in under 0.2 seconds.

Algorithm performance on sparse-unit-bpm (Fig. 12). Dinitz and Tidal flow were the only algorithms that could complete any of the tests. Dinitz performs slightly better than Tidal Flow on this entire test suite.

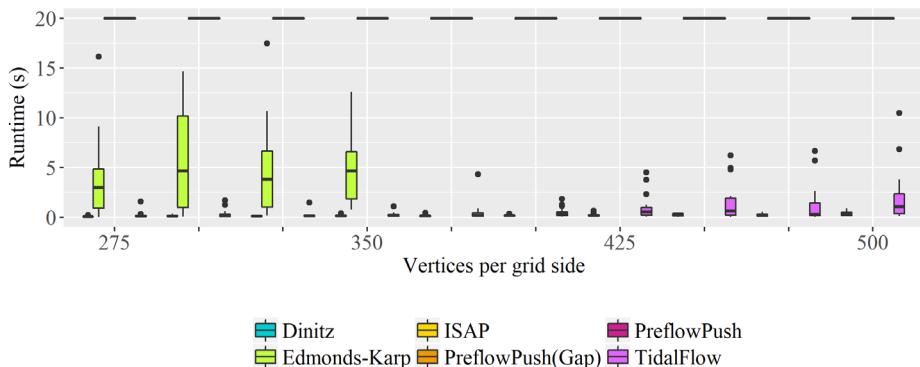


Fig. 13. Algorithm performance on grid.

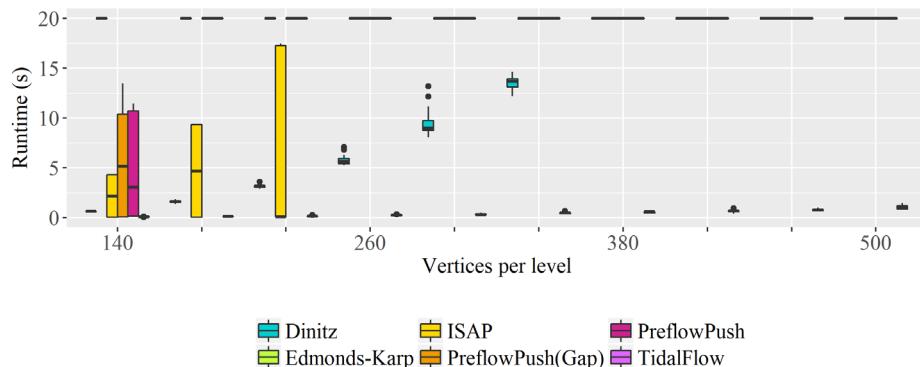


Fig. 14. Algorithm performance on level-10.

Algorithm performance on grid (Fig. 13). Dinitz manages to complete all tests in under 1 second. Edmonds-Karp completes up to size 350 before timing out. TidalFlow's average case is comparable to Dinitz but has several cases much slower.

Algorithm performance on level-10 (Fig. 14). Edmonds-Karp fails to complete any graphs. Dinitz completes graphs up to size 340 before timing out. ISAP can complete graphs up to size 220. PreflowPush(Gap) completes only graphs size 140. Preflow-Push also completes only graphs of size 140. Tidal Flow completes all graphs in less than 2 seconds.

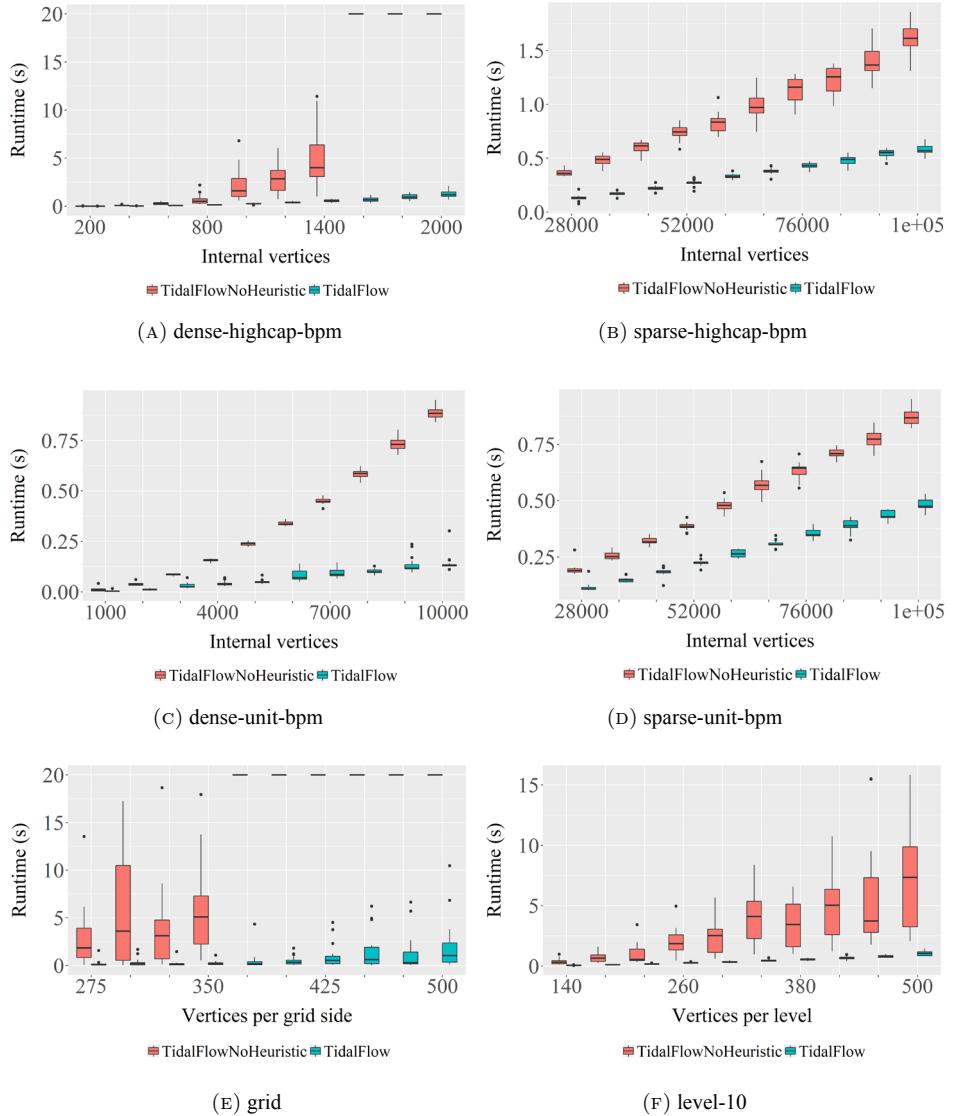


Fig. 15. Running Tidal Flow without the heuristic function h .

In Fig. 15 Tidal Flow using the heuristic function h is compared against a different implementation of Tidal Flow that removes the heuristic function. For every class of graphs, removing the heuristic function resulted in a significantly slower algorithm. The algorithm also started to have a wider variance in running time.

6. Discussion

6.1. Comparison to other Fast Flow Algorithms.

Tidal Flow performed well compared to other flow algorithms. Perhaps the most surprising performance was the behavior of Preflow-Push algorithms on the generated graphs. In Ahuja's study (Ahuja *et al.*, 1997) Preflow-Push algorithms perform far better than other algorithms. This is most likely due to Preflow-Push being implemented without the global relabeling heuristic. That heuristic seems to make a huge difference in the behavior of Push-Relabel. In this experiment Dinitz's algorithm significantly outperformed ISAP. In contrast, Ahuja's study found the two algorithms to be comparable. The improvements in Dinitz's algorithm's performance are likely due to the implementation improvements suggested in Dinitz (2006).

Dinitz's algorithm performed better than Tidal Flow on unit capacity bipartite matching cases. Dinitz's algorithm has a running time of $O(m\sqrt{n})$ on this class of networks and generates blocking flows in $O(m)$ time. Surprisingly, Tidal Flow is not that much slower than Dinitz's algorithm on these cases. Tidal Flow outperformed Dinitz's algorithm on dense level graphs and dense bipartite graphs with large edge capacities.

6.2. Importance of the Heuristic Function.

Two versions of Tidal Flow were implemented for the purpose of measuring the importance of the heuristic function h . In one implementation, the heuristic function h was removed. In every graph class, Tidal Flow performed significantly worse without the heuristic function. The effect of the heuristic function is to help identify bottlenecks in the network. When the heuristic function is tight, more flow can be sent down other paths in the network during low tide. The importance of the heuristic function was most pronounced in the grid network. This behavior is most likely due to the fact that level graphs formed from the grid network have a large amount of separating and rejoining paths. The heuristic function provides reasonable guidance so that less flow gets stuck in the middle of the network during low tide on such networks.

7. Concluding Remarks

This paper introduced the Tidal Flow algorithm and gave a preliminary survey measuring the performance against other flow algorithms. Tidal flow is both simpler to understand and implement than other fast flow algorithms. The relationship to preflows makes Tidal Flow a good intermediate algorithm for understanding more complicated algorithms like Preflow-Push. Though this paper described an initial exploration of the algorithm, there are still a number of unknowns with respect to the performance of Tidal Flow. Though Tidal Flow performs well on random networks against other flow algorithms, more extensive testing is required to determine its worst case behavior. Additionally, the theoretical worst case running time of $O(nm^2)$ may not be tight. Whether it is possible to create a level graph requiring more than $O(n/\log(n))$ tide cycles to block the network is also unknown. Finally, it is unclear if a better h function exists for guiding the low tide decisions.

8. Acknowledgements

The author is grateful to Antony Stabile and Travis Meade for their contributions in developing and simplifying the initial version of Tidal Flow, the students of the UCF Programming Team for the continued feedback on how Tidal Flow is taught, and Lisa Soros and Brian Dean for feedback on a preliminary version of this paper.

References

- Ahuja, R.K., Magnanti, T.L., Orlin, J.B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Boykov, Y., Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9), 1124–1137.
- Cherkassy, B.V., Goldberg, A.V. (1995). On implementing push-relabel method for the maximum flow problem. In: *Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization*. London, UK, UK. Springer-Verlag, 157–171.
- Dinitz, Y.A. (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation. *Doklady Akademii Nauk SSSR*, 11, 1277–1280.
- Dinitz, Y.A. (2006). Dinitz's algorithm: The original and even's version. In: *Theoretical Computer Science: Essays in Memory of Shimon Even*, volume 3895. Springer, Berlin, Heidelberg, 218–240.
- Edmonds, J., Karp, R.M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2), 248–264.
- Ford, L.R., Fulkerson, D.R. (1955). Maximal flow through a network. *Canadian Journal of Mathematics*, 8, 399–404.
- Forišek, M., Stienová, M. (2013). *Explaining Algorithms Using Metaphors*. Springer, New York, NY, 1 edition.
- Goldberg, A.V., Hed, S., Kaplan, H., Tarjan, R.E., Werneck, R.F. (2011). Maximum flows by incremental breadth-first search. In: Demetrescu, C. and Halldórsson, M. M. (Eds.), *Algorithms – ESA 2011*. Berlin, Heidelberg. Springer Berlin Heidelberg, 457–468.

- Goldberg, A.V., Tarjan, R.E. (1988). A new approach to the maximum-flow problem. *J. ACM*, 35(4), 921–940.
- Goldberg, A.V., Tarjan, R.E. (2014). Efficient maximum flow algorithms. *Communications of the ACM*, 57(8), 82–89.
- Harris, T. E., Ross, F. (1955). Fundamentals of a method for evaluating rail net capacities. Technical report, Rand Corporation, Santa Monica, CA.
- Ahuja, R.K., Kodialam, M., Mishra, A.K., Orlin, J. (1997). Computational investigations of maximum flow algorithms. 97, 509–542.
- Karzanov, A.V. (1974). Determining a maximal flow in a network by the method of preflows. *Doklady Akademii Nauk SSSR*, 15(2), 434–437.
- Lakoff, G., Johnson, M. (2003). *Metaphors We Live By*. University of Chicago press, Chicago, IL.
- Lerner, J., Wagner, D., Zweig, K.A. (Eds.) (2009). *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation*. Springer-Verlag, Berlin, Heidelberg.
- Orlin, J.B. (2013). Max flows in $O(nm)$ time, or better. In: *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC’13. New York, NY, USA. ACM, 765–774.
- Orlin, J.B., Ahuja, R.K. (1987). *New Distance-Directed Algorithms for Maximum Flow and Parametric Maximum Flow Problems*, Technical Report. MIT, Cambridge, MA.
- Sedgewick, R., Wayne, K. (2011). *Algorithms*. Addison-Wesley Professional, 4th edition.



M. Fontaine is the host of algorithms YouTube talk show, Algorithms Live! He is the cow artist for USA Computing Olympiad, creating the t-shirt design since 2014 and has been a USACO coach since 2017. He was an instructor at the University of Central Florida from 2014 to 2017 and a coach of UCF’s ACM ICPC team from 2013 to 2017. He received his M.S. from UCF and was a research assistant at UCF’s Institute for Simulation and Training from 2008 to 2013 studying problems in distributed game-based training.

Algorithmic Cognition and Pencil-Paper Tasks

David GINAT

*Tel-Aviv University, Science Education Department
Ramat Aviv, Tel-Aviv, Israel 69978
e-mail: ginat@post.tau.ac.il*

Abstract. Pencil-paper algorithmics was displayed by several IOI studies, which examined various structural and scientific characteristics of offered tasks, including “what makes a good task”. We offer an additional facet, of cognitive considerations. We underline the aspects of abstraction, heuristics, creativity, and declarative conceptions, which are relevant already with pencil-paper algorithmics. We describe the settings of our country’s Stage-A, pencil-paper exam, and display cognitive considerations of the exam tasks. We illustrate these considerations with several single-input tasks, in which hidden patterns should be recognized for effective “computation by hand”.

Keywords: algorithmic problem solving, cognitive aspects.

1. Introduction

Consider the following Reversing algorithmic task. Given the following sequence of twenty eight A/B characters: A B B B A B B A A B A A B A A B A B A A A B B A B B B A, what is the minimal number of reverse-sub-sequence operations necessary for obtaining the alternating sequence A B A B ... A B? (A Reverse-sub-sequence operation may be applied on any sub-sequence of consecutive characters; e.g. if applied on the first 4 characters A B B B it will yield B B B A.)

The task specification is short. It requires no special knowledge and involves a simple operation, to be repeatedly applied on a particular input. The input is not long, but still long enough so that one will have to reason in an ordered analysis, which will yield an operational computation from the starting point (the input) to the desired goal. Due to the limited size of the input, the operational computation can be performed by hand, without a computer.

More can be said. In solving the task, one should elevate the point of view and focus on particular task elements, while ignoring others. This involves *abstraction*. One should also cleverly process these elements. This involves *creative* employment of *heuristics*. And one should convince herself (even if intuitively) of optimality. This involves *declarative* conceptions. The cognitive aspects of abstraction, heuristics, creativity, and

declarative conceptions are essential in algorithmic problem solving. In this paper, we relate them to the preliminary pencil-paper exam in algorithmics.

Previous IOI-related studies of pencil-paper algorithmics display and discuss tasks to be solved at home or in an actual on-site exam (e.g., Burton, 2010; Kubica and Radoszewski, 2010; van der Vegt, 2012; Radoszewski, 2014). Additional studies display a variety of related forms of non-programming activities (e.g., Dagiéne, 2006; Dagiéne and Futschek, 2008; Opmanis, 2009). These studies describe contest settings and offer task considerations, including those that “make good tasks”. The considerations focus on task features of structural and scientific characteristics (including time frames, tools used, mathematical and algorithmic features, sub-tasks and more). In this paper, we relate to these characteristics, and add considerations of algorithmic cognition.

We introduce and discuss an approach of preliminary pencil-paper algorithmics, which comprises the initial OI activity in Israel. In the next section we motivate and describe the considerations underlying our approach. In the section that follows we illustrate the approach with different examples, additional to the Reversing task. In the last section we mention our implementation experience and enquire about the correlation and differences between mathematical competence and algorithmic competence.

2. Considerations of Pencil-Paper Algorithmics

We display below setting considerations and cognitive consideration that yield our approach of pencil-paper problem solving in algorithmics.

Setting Considerations

- **Wide population.** In Israel, a limited amount of students study high school computer science (CS), from 10-th grade. In the initial IOI activity we try to reach as many motivated students as possible, including young mathematics students who are not necessarily acquainted with programming.
- **Two-fold goal.** We aim at promoting interest in algorithmic challenges, as well as identifying competent algorithmics students. About 3000 students, nation-wide, show interest in our Stage-A.
- **National 2-hour exam.** We announce a national Stage-A exam in the beginning of the academic year. The exam takes place in the schools, under the supervision of teachers who print the exam just before it starts.
- **Teacher involvement.** We encourage teachers to take part, even if small. We need the teachers for encouraging students to prepare-for, and attend the exam.
- **Student preparation.** Students should have an idea of the exam format. We display in our website exams of previous years (from 2010).
- **Four/five different exam tasks.** We estimate 15 to 45 minutes for a task, in the 2-hour exam. The tasks are ordered according to their levels of difficulty. The first one is rather simple, so that students will feel that they managed to solve at least one task. Some questions are optimization questions, some involve a combinatorial computation, some display a two-player game and ask for the first

move, some are related to generic computational schemes, and some just require logical reasoning.

- **Short and simple specifications.** Reading and understanding a task takes time. We try to minimize this time. Task specifications are very short, often with a short illustration, and with a single input on which to perform a computation.
- **Colourful challenges.** The tasks pose challenge. Usually, there is no story in them, but students find the tasks colourful, due to their challenge.
- **No programming knowledge.** One may do well without programming background. In addition, programming knowledge in an early age often involves a lot of technical details, and this may not help much here. What may sometimes help is prior experience with problem solving.
- **Single, non-trivial input.** The vast majority of the tasks involve a single input. The input is in a size that requires insight and competence. One may sometimes guess an answer, but this is very unlikely to succeed with a set of tasks. When insight is obtained, a pencil-paper computation may be performed in a reasonable time. If no insight is obtained, the computation may take long time, or not take place at all.
- **Single integer output.** The answers of the few thousand students are submitted electronically. Due to the large amount of data, the task outputs are very short – usually a single integer per task. We believe that it is sufficient for evaluation.
- **Hints for checking the output.** Once insight is gained, a careful computation should yield the right result. Still, in order to help avoiding erroneous calculations, we provide hints, such as “The digits unit is odd” or “The output is a multiple of 5”.
- **Partial credit.** Since a task answer is a single-integer, there is usually no partial credit. Yet, there are exceptions. Occasionally, upon the electronic checking we notice that a group of students obtained the same result, which is different from the correct one. We figure out the rationale for this result, and if we realize that it may have been obtained from partial recognition of patterns, we give partial credit.
- **Passing criteria.** We choose about 300 students, out of up to 3,000 (the number varies in different years) based on their performance with the more challenging questions. These students are invited to a more thorough Stage-B exam.

Cognitive Considerations

Although the tasks are single-input/single-output tasks, solved with pencil and paper, we regard them as reflecting cognitive competencies that are essential for algorithmic problem solving. We believe that problem solvers should dedicate non-negligible time for solving a challenge. In a 2-hour exam, with 4 tasks (and perhaps a bonus one), one may have about 30 minutes per task, on the average. This may be sufficient, for a competent student in a pencil-paper stage, for gaining insight and capitalizing on it. In this amount of time one may employ relevant cognitive faculties.

- **Abstraction.** Algorithmic problem solving involves abstraction (e.g., Wing, 2006; Armoni *et al.*, 2006; Ginat and Blau, 2017). Abstraction may be expressed in a variety of forms. One may notice mapping (reduction) from a given question to another; or offer an illuminating representation that considerably simplifies the viewpoint on a given task; or focus on particular elements while ignoring others. In

looking at the Reversing task, one should momentarily ignore the inner characters of a reversed sub-sequence and focus only on its ends, in order to notice the asset of concurrently “breaking” AA’s and BB’s.

- **Heuristics and reasoning.** Challenging tasks are solved by employing various kinds of heuristics, such as problem decomposition, backward reasoning, generalization, and more (Polya, 1945; Schoenfeld, 1992). Rigorous reasoning and case analysis are combined with the application of heuristics. Hidden patterns are unfolded. For example, in the Reversing task, careful reasoning/analysis may decompose the input disorders into two cases – the case of AA’s and BB’s and the case where the ends are improper (e.g., an A in the right end). A single `reverse` operation may concurrently “break” an AA and a BB. How should a disordered end be handled? Creativity may help here.
- **Creativity.** Solution processes require divergent thinking (e.g., Ginat, 2008). In doing so one may need to examine several solution directions, demonstrate flexible associations, and invoke original ideas. In attempting the case of disordered ends in the Reversing task, one may combine flexibility with the heuristic of auxiliary construction, and add an auxiliary A to the right end of the sequence. This will transform the end case into an AA/BB case.
- **Declarative perspective.** Algorithmic problem solvers naturally turn to operational reasoning, and go for the “how” computation. Yet, an operative perspective may be insufficient when one wants to be convinced of correctness and efficiency (e.g., Ginat, 2008). For this, one needs to see the declarative meaning, even if not formally, for believing correctness. In the Reversing task, in order to be convinced of optimality, one should notice that after the auxiliary construction (of an A added to the right end), the number of AA’s is exactly equal to the number of BB’s, and a single `reverse` operation may reduce at most 1 of each.

In the next section we demonstrate the above elements with additional tasks of different types, which were posed in our Stage-A exams during the last five years.

3. Illustrations

In the previous section we exemplified cognitive considerations with the Reversing task. We regard the Reversing task as relatively easy. It was one of the first two questions (out of four) in the 2012/13 Stage-A exam. The patterns to recognize are not immediate, but also not very challenging. The following is an additional task of limited challenge.

Sum of sub-sequences. In the following sequence of integers: 4 11 3 5 3 there are 4 sub-sequences of consecutive integers whose sums are multiples of 3. These are the sub-sequences: 4 11; 3 (the left 3); 4 11 3; 3 (the right 3). Notice that a single integer is regarded as a sub-sequence. So is the whole sequence. Given the following list of integers, output the total number of sub-sequences whose sums are multiples of 3?

6 1 4 124 3 6 512 3 1 33 2 2 32 100 813 4 41 1 8 213 5 7 61 8 42 1 4 2 20 8

Hint: the total sum is a multiple of 5.

This task was one of the first two tasks of the Stage-A exam of 2014/15. The challenge here is to devise a simple scheme that offers an ordered way of counting. Problem solvers may attempt various types of counting here. They may be based on two observations: **1.** Counting is simplified by examining remainders of 3; and **2.** An ordered counting may be carried out with a single pass over the input in which, for each integer – the number of sub-sequences that it “ends” will be added to a total sum. We may specify a corresponding declarative notion.

The number of sums that are multiples of 3, which an integer v ends, is equal to: the number of integers to the left of v in the input, for which the mod-3 remainder of the sum from the left-end to each of them equals the mod-3 remainder of the sum from the left-end to v .

One does not need to specify the above notion explicitly, but should be able to see it (or an equivalent one) in order to apply an ordered operational computation with pencil and paper. The counting involves a feature of working backwards and an abstraction aspect in which the relevant sums are distinguished from all the possible sums. The hint may help avoiding calculation mistakes. A student that will not gain corresponding insight will face difficulties, and may spend a long time on the task.

* * *

The next task involves the two-player game of Chomp. This game was also posed in the Australian Informatics Competition with a 3×3 chocolate block (Burton, 2010). We displayed it in our Stage-A of 2017/18 in a different form.

Board game. Given a board of $N \times M$ squares, two players play against each other. Each player on her turn marks one of the rectangle squares. As a result, this square, and all the squares to its right and/or above are removed. The game ends when no squares remain. The player who makes the last move loses the game. If, for example, in the 3×4 board below the square F is marked, then the squares B, C, D, F, G, H will be removed. We assume that each player plays the best she can.

A	B	C	D
E	F	G	H
I	J	K	L

A. In the 2×10 board below, the first player will win the game if she marks in her first move one particular square. Which square should she mark?

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T

B. Answer the same question for the upper 3×4 board.

We ask about the first move in a 2-player game in some of our Stage-A exams. Students who prepare, and look at previous exams should be ready for such questions. In part A one should examine small rectangles, such as 2×2 , 2×3 , and 2×4 boards, and

generalize. The heuristic of generalization from simple cases will yield the following declarative, invariant pattern for winning as the first player.

After every move of mine, the top line will be a square shorter than the bottom line.

One does not need to specify the invariant explicitly, but should see the pattern it involves. In part B, the board is smaller, and no generalization is needed. But thorough case analysis should be carried out. All in all, each part involves somewhat different competencies. In our experience, students felt enthusiastic about this task, though it was not trivial, and not fully solved by many.

* * *

The next task involves a combinatorial computation that may be solved in various ways. Combinatorial computations are common in algorithmic problem solving. They appear in various forms in some of our Stage-A exams. The following task was designed by our coaching team member Daniel Hadas, for Stage-A of 2017/18.

Watch colouring. Given a round watch with the integers 1..12, and four different colours; what is the number of different ways to colour the 12 integers so that every two adjacent integers will be coloured with different colours, and every two opposite integers (e.g., 5 and 11) will be coloured with the same colour? Hint: the units digit of the answer is 2.

This task is harder than the previous ones. A student who studied combinatorics may have some advantage, but the solution will not be immediate. One may try diverse ways of case analysis, but should be careful not to count some colouring twice or miss a colouring. In our view, one needs to demonstrate creativity with suitable heuristics in order to simplify as much as possible the view of the task.

Since the colour of every two opposite integers in the clock is the same, we may reduce the task to the colouring of 6 integers in a circle. Counting would have been much simpler if the integers were in a line and not a circle, since a line has two explicit ends. The challenge here stems from the need to avoid colouring the two ends, 1 and 6 in the same colour. How should we do that? A creative embedment of the notion of “complement” in the heuristic of decomposition paves the way. We specify the relevant observation in an operative (rather than declarative) manner.

Rather than adding-up all the legal colouring cases, count all the ways to colour 1 to 6 in a line, and then remove the ways in which the colours of 1 and 6 are the same; the number of removed ways is exactly the number of all the circular colourings of 1 to 5, as we may view 1 and 6 as one unit, with one colour.

The notion of “complement” appears here with the recognition that the number of legal circular colourings of 1..5 **complements** the number of legal circular colourings of 1..6 to the number of line colourings of 1..6. Thus, a “line case” may be viewed as being composed, number-wise, of two different “circular cases”.

At this point, one should realize that in order to answer the case of “circular 1 to 5”, one needs to know the solution of “circular 1 to 4”; which requires the solution of “circular 1 to 3”; and so on. This observation involves the heuristic of backward reasoning. The pencil-paper calculation should be built bottom-up, in reverse to the backward reasoning analysis.

All in all, this colourful challenge requires both of the heuristics of decomposition and backward reasoning, and an elegant creative invocation of the notion of “complement”.

* * *

Our next example is different from the previous ones. It asks a question about the execution of a given algorithmic process. The question is about a particular state that will be reached after a given amount of time. Algorithmic problem solvers need not only design an algorithmic solution but also comprehend a given one. The question is an extension of an old challenge (about ants), which we assumed to be unfamiliar to the students. Since the question is somewhat different from others in our Stage-A exam, we posed it as a bonus, fifth question (in the exam of 2015/16).

Balls on a track. Eleven identical balls are spread on a 100 cm track. The two ends of the track – location 0 and location 100 are blocked with barriers. Each ball is put initially in a location indicated (in cm) by an integer below. The balls start moving at time 0, each in the direction indicated (near its location) below.

|| ← 6 14 → ← 24 ← 38 44 → ← 50 ← 54 64 → ← 74 82 → 88 → ||

Each ball moves in a constant velocity of 1 cm per second. When a ball collides with a barrier in one of the ends, or when it collides with another ball, it switches its moving direction (and continues to move in the same velocity). For example, the balls that are initially in locations 44 and 50 will collide 3 seconds from the start, and will switch their moving directions. What will be the location of the ball that is initially fourth from the left (in location 38) after 5 minutes from the beginning?

The solution of this task requires abstraction. An examination of the movements of explicit balls yields a cumbersome, probably impossible pencil-paper computation. One may do much better with an alternative perspective, in which some details are disregarded.

Since the balls are identical, and their velocities are the same, one may regard each ball as anonymous and *overlook* the ball collisions. This viewpoint enables a view of a collision between two balls as an event that does not have any impact on the movements of the two identical balls in their original directions. This abstract point of view considerably simplifies the view of the task.

In addition, one may further extend this train of thought to disregard collisions with the right and the left ends. One may *pretend* that there are no left-end and right-end, and extend the track with a sequence of its copies in each direction. This allows one to view each ball as moving steadily in its original direction for 300 seconds. Once one obtains the final location of each ball in the sequence of copies, one may return to the original task, and transform this location into a concrete location in the given track. Since the balls remain in their original order, the answer would be the location that will be fourth from the left.

All in all, they key feature of the solution is abstraction, in which one does not view the task in its original arrangement, but rather as one with an arrangement yielded from an “as if” perspective (Ginat, 2010), of “anonymous” balls that collide neither with each other nor with the barriers.

3. Discussion

The examples presented in this paper display tasks with diverse characteristics that may be relevant for a preliminary stage of the OI activity. Elements that were embedded in the examples include: algorithmic design with a repeatedly used operator, summation schemes, game instances, invariance, the notion of complement, case analysis, logical reasoning, algorithmic tracing, and more. Such elements are apparent in algorithmic problem solving.

Yet, the features that we tried to underline are related to essential cognitive aspects involved in algorithmic problem solving; in particular those of abstraction, heuristics employment, creativity, and declarative conceptions. The relevance of these aspects was shown here with pencil and paper tasks.

We exemplified diverse appearances of these aspects with several illustrations. Abstraction was exemplified in the first and second tasks (Reversing and Sum of subsequences) with focusing on particular elements and ignoring others. It also appeared in the fifth task (Balls on a track) with the notion of “as if”, which involved a change of perspectives. The employment of heuristics appeared in all the tasks. Particular heuristics that were relevant included problem decomposition, auxiliary construction, generalization from simple cases, and backward reasoning. Creativity appeared in two of the tasks – creative auxiliary construction in the Reversing task, and creative decomposition, using the notion of “complement” in the fourth, Watch colouring task. Declarative conceptions were relevant in all the tasks. It was particularly apparent in observing optimality in the Reversing task and in recognizing an invariant pattern in the third, Board game task.

Our experience with the presented tasks (posed in different years) show that the first two were solved by about a third of the students, the third – by about a fifth of the students, the fourth – by less than a tenth of the students and the fifth – by an even smaller number of students. We noticed that many students spent a long time on the simpler tasks, and often obtained only partial insight. They were then left with little time for the harder tasks.

Nevertheless, students were enthusiastic about the tasks. Many spent extra time after the exam to solve the tasks that they did not manage to solve during the exam. Some of the teachers were enthusiastic as well, although several of them felt uncomfortable, since they could not solve the tasks themselves. Interestingly, some of the students invited to the next stage did not attend it, and said that they attended the Stage-A exam “just for the challenge” of coping with colourful tasks.

Each year, after checking the answers we choose about one tenth of the students (250–300 students) for the next stage, according to the tasks they solved. About 15% of those invited to the next stage are girls. We pay particular attention to students that do really well in the Stage-A exam. Quite a few remain at the top in the next stages.

Some of the top students are also very competent in our country's IMO activity. We wonder about the correlation between success in the IOI and success in the IMO. Obviously, there is a correlation, but there are also differences.

Mathematical thinking and algorithmic thinking involve the recognition of hidden patterns, which is strongly tied to declarative conceptions and the notion of “knowing that” (Ryle, 1949). Algorithmic thinking also involves a strong facet of “knowing how”. The observations sought by algorithmic problem solvers should yield suitable operational computations. Competent IOI problem solvers effectively combine the “that” and the “how”, and usually turn to the “how” only after seeing the “that”. For competent IMO problem solvers seeing the “that” may often be sufficient.

At the preliminary level of pencil-paper algorithmics we may not expect involved operational computations. However, it is still relevant to aim for an operational computation that will be carried out only after hidden patters are recognized. The cognitive aspects underlined here illuminated a facet of recognized “that” that paved the way to an operational “how”. Suitable awareness of these aspects may assist task designers in their designed and posed tasks, including pencil-paper tasks.

Acknowledgement

We thank Hanit Galili and Nir Lavee, from the Israel IOI team, for statistical information about students' success in our Stage-A tasks.

References

- Armoni, M., Gal-Ezer, J., Hazzan, O. (2006). Reductive thinking in computer science. *Computer Science Education*, 16(4), 281–301.
- Burton, B. (2010). Encouraging algorithmic thinking without computer. *Olympiads in Informatics*, 4, 3–14.
- Dagienė, V. (2006). Information technology contests – introduction to computer science in an attractive way. *Informatics in Education*, 5(1), 37–46.
- Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R.T. and Syslo, M.M. (Eds.) *Informatics Education – Supporting Educational Thinking, Lecture Notes in Computer Science*, 5090. Springer, 19–30.
- Ginat, D. (2008). Learning from wrong and creative algorithm design. *Proc of the 39th ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 26–30.

- Ginat, D. (2010). The baffling CS notions of “as-if” and “don’t care”. *Proc of the 41st ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 385–389.
- Ginat, D., Blau, Y. (2017). Multiple levels of abstraction in algorithmic problem solving. *Proc of the 48th ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 237–242.
- Kubica, M., Radoszewski, J. (2010). Algorithms without programming. *Olympiads in Informatics*, 4, 52–66.
- Opmanis, M. (2009). Math contests: solutions without solving. *Olympiads in Informatics*, 9, 147–161.
- Polya, G. (1954). *How to Solve it*. Princeton University Press.
- Radoszewski, J. (2014). More algorithms without programming. *Olympiads in Informatics*, 8, 157–168.
- Ryle, G. (1949). *The Concept of Mind*. The University of Chicago Press.
- Schoenfeld, A.H. (1992). Learning to think mathematically: problem solving, metacognition, and sense making in mathematics. In: Grouws D.A. (Ed.), *Handbook of Research on Mathematics Teaching and Learning*. 334–370.
- van der Vegt, W. (2012). Theoretical tasks on algorithms; two small examples. *Olympiads in Informatics*, 6, 212–217.
- Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.



D. Ginat – heads the Israel IOI project since 1997. 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.

New Approach for Comparison of Countries' Achievements in Science Olympiads

Mile JOVANOV¹, Marija MIHOVA¹, Bojan KOSTADINOV²,
Emil STANKOV¹

¹*Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University
st. Rugjer Boskovicj 16 Skopje, Macedonia*

²*Cloud Solutions LLC*

st. Jurij Gagarin 33/27 Skopje, Macedonia

*e-mail: mile.jovanov@gmail.com, marija.mihova@finki.ukim.mk,
bojankostadinov@gmail.com, emil.stankov@gmail.com*

Abstract. There are several International Olympiads for secondary school students (for example, mathematics, physics, chemistry, biology and informatics). These Olympiads are not just a science competition, but a means to care for talent in the particular science.

The goal of this paper is to identify the necessary topics important for good results at these international contests, and to compare the contest systems for the countries in South Eastern Europe, in the field of Informatics (Computer Science), as a region that is one of the prominent world regions in the context of high results in the international competitions. Here, we provide comparison through detailed analysis of several countries, and further we present a new approach that may be used to compare the achievements of the countries based on the results that students achieved at these competitions. Finally, we present an application of this approach on the results of some of the discussed countries compared to Macedonia.

We strongly believe that the paper will provide a valuable content and approach for the entities involved in the organization of the contests, to measure their results compared to other countries, to use the information for improvement, and to use their achievements to raise awareness among the government institutions and companies in order to get support from them.

Keywords: students, STEM education, science competitions, programming.

1. Introduction

There are several International Olympiads for secondary school students, but five of them that are most widely recognized are those in mathematics, physics, chemistry, biology and informatics. Most of these were originally founded under the auspices of UNESCO (Taylor, 2012). These Olympiads are not just a science competition, but a means to care for talent in the particular science. Caring for talent involves a broad range of issues, including identification of talent and education adjusted to that talent (Verhoeff, 2011).

In particular, the International Olympiad in Informatics (IOI, 1989) is an annual programming competition, reserved for secondary school students. Students compete by solving problems (of algorithmic nature) in one of several available programming languages (C, C++, Pascal, and, as of 2015 – Java). In the last few years, participants received full feedback for their submissions, which wasn't the case with earlier Olympiads. The contest is usually organized in two competition days, and countries are limited to 4 competitors each (Kostadinov *et al.*, 2015). The first IOI was held in 1989, and this year, the thirtieth Olympiad will be held (in Tsukuba, Japan).

In general, competitions are a major factor in education (Verhoeff, 1997). A lot of countries use different forms of competitions in order to encourage students to perform better in school (for example, in order to earn scholarships or grants), to bring the best out of them (by enrolling students into special programs based on their interests), or to promote cooperation by grouping students into teams and teaching them (through competitions) the importance of collaboration between people with different talents and preferences (Kostadinov *et al.*, 2015). Whatever the goal, competitions must be fair in order to make them engaging for the participants. That is, each participant (student, team) should have the same chances, as possible, of winning. All science Olympiads try to satisfy this criterion.

The role of competitions in informatics is additionally emphasized in education, since the current status of informatics education is unsatisfactory in many countries (Guerra *et al.*, 2012). Although computers, applications, and information technology (IT) in general are an increasingly natural part of the everyday work in schools, focus is mainly put on basic digital literacy skills, while the underlying principles are left uncovered. This situation has been recognized as a problem in many countries, and recently the introduction of computing in the curriculum (e.g. UK, Estonia) has resulted in an increased debate throughout Europe (Dagienė, 2014). Macedonia is one of the first countries that has also included Computing as a compulsory subject in the primary school curriculum (Jovanov *et al.*, 2016).

Having in mind the importance of competitions, especially on international levels, each country has a specific educational system and different attitude and focus on them. This depends on many factors that include the base of teachers in the particular field, their dedication, as well as the dedication of the state institutions on the promotion of the competitions and their support. In some countries the main effort for the conduction of competitions is undertaken by private or non-government organizations or even by a few individuals.

In this paper, we will explain the strategies for good results at IOI (and IOI-like regional contests) in the next section, and then we will focus on the organization of the informatics competitions in selected countries from the South Eastern Europe (SEE) region, as a region that is one of the world's prominent regions in the context of high results in the international competitions. The analysis has been done through extensive interviews with relevant representatives from Romania, Bulgaria, Serbia, Montenegro, Slovenia and Turkey, who are involved in the organization of their competition cycles. Additionally, we include the situation in Macedonia, from our relevant experience. The

findings will be summarized in a table that will allow quick comparison that may be used for further actions on the improvement of the effort in some country. This will be given in the following section.

Further, in Section 4, we will present a new approach that may be used to compare the achievements of the countries based on the results that students have achieved at these competitions, and in Section 5 we present an application of this approach on the results of the discussed countries, especially compared to Macedonia. At the end we present the conclusions.

2. Achieving a Good Result at an International Olympiad

To win a medal at any of the international science Olympiads, the student should really have great qualities, but that's not enough. ***Our experience in participating in competitions and recruiting successful competitors shows that several factors are important: quality education, gift, tradition, competitive environment, continuous work and quality training.***

For example, Macedonia has a much smaller population than most countries, and the number of gifted students is statistically much lower, so in order for a student to enter in the rank of medals at the international competitions, a serious commitment to each of these aspects is required. In the following subsections we will give our summarized knowledge and experience in the particular topics.

2.1. *Quality Education*

The educational system in one country plays a major role in its success at the major competitions in science. The first introduction to the basics of a particular area usually happens in schools. It is the place where the student realizes that he has the predisposition to understand the subject. The encouragement of students who are interested in some discipline is firstly made by their teachers in regular education and they are the ones who can judge whether a particular student stands out from the rest of the students in the given area. They are the first to give guidance to these students, to encourage additional activities and to develop their love for science.

2.2. *Gift*

The most important thing that a student should possess to be able to win a medal at an international Olympiad is an extremely great gift for science. According to the authors of one of the most prestigious books in algorithms (Cormen *et al.*, 2009), to design and prove an algorithm is an art, so they call the development of algorithms "the art

of designing algorithms”. This is an important element especially when it comes to competitions that happen in a short period of life, such as programming contests, in which students have only 2–4 years to reach some level of knowledge that is needed for success. Therefore, here the student must be able to quickly learn new methods and techniques, to clearly perceive the problem in order to design a solution and to handle unforeseen situations easily. The scientific committees of international Olympiads, such as the IOI and IMO, prefer to separate students who possess extremely high logical intelligence rather than identify those who have learned the most number of techniques. They try to make original tasks that can be solved without the use of great theoretical knowledge.

2.3. Tradition

Countries with long-standing tradition achieve greater successes in international competitions, regardless of economic and political events or their size. A proof to this claim are the relatively small countries such as Bulgaria, Serbia and some former Russian republics, which have also won gold medals. Each new generation learns from the previous one and tries to surpass it, and even if it fails – the quality level cannot fall significantly, first of all because the standards are set to a relatively high level. Knowledge from an area in a given country is “a continuous smooth function”, which gradually changes over time, so it is necessary to spend a few years for a greater decline or growth. And our results show just that.

2.4. Competitive Environment

Rarely can a student greatly differ in his knowledge and skills as compared to the students in his environment. Every student compares and competes first with the students of his own school, his city, and finally – with the students of his country. For each student, firstly it is most important to be selected for the international Olympiad team, and only after to achieve significant success at the Olympiad. So, when a student is aware that there is no competition in his/her environment, he/she will work much less. Hence, it is extremely important to create a larger group of students capable of achieving a good result in national competitions, so that they motivate each other to overcome one another.

2.5. Continuous Work

As in any other area, giftedness without work will not produce results. To win a medal at the Olympiads, one must work all year long. In informatics, students have the opportunity to participate in online contests organized worldwide.

2.6. Strong Training

One of the negative things about on-line competitions is that they last shorter than the contests at the Olympiads, and that some of the problems that need to be solved there are considerably easier. Sites rank the competitors, so it is important for them to solve as many tasks as possible. Therefore, organized preparations (training camps) should aim, for example, to change the students' approach of solving easier tasks in a shorter amount of time, to solving a difficult task in a longer amount of time. Of course, they also must get them to improve their programming skills.

2.7. Specifics for the Macedonian Case

In this subsection we will point out some specifics from our personal experience in Macedonia relevant for the topics above, which may put an additional light on the subject.

Our experience shows that there is a small number of teachers in Macedonia who want to work with their students additionally, who try to teach them above the standard curriculum and who are willing to participate in competitions and other events with their students. But does the problem lie only in the quality and commitment of teachers? In recent years, the government has introduced several measures that should encourage teachers to work more actively with gifted students, which include (financial) prizes for the results achieved at international competitions. But, the measures are not "attacking" the problem systematically, so we still lack results.

The lack of training in the regular school for informatics competitions transfers this problem to the Computer Society of Macedonia (CSM), as a non-government institution responsible for the participation at international Olympiads. Therefore, the trainings that CSM organizes include not only students that are selected for the international contests but also some other students who have shown great results in the local competitions (close results to those of the selected students). Also, CSM tries to send students to other international competitions outside Macedonia, including students who are not in the first team. The trainings that are organized by CSM are really strenuous and can last more than 10 hours per day.

In the following section we will present specifics for selected countries of the SEE region.

3. Current Situations in Selected Countries from SEE

In this section we will present our findings, based on reports, interviews and published papers, for the situation in selected countries.

Information regarding the situation with competitions in **Serbia** was provided by Jelena Hadzi-Puric, a professor at the Faculty of Mathematics, University of Belgrade.

The annual cycle of Serbian competitions consists of four levels: Municipal or online Qualifications, Regional, National Competition, and the Serbian Olympiad in Informatics (SIO). This year, around 750 junior contestants have been involved in the first level qualifications, and 1273 senior contestants tried to solve at least one problem. For the Regional and National competitions, students are divided into two divisions: A and B. Division A consists of contestants from schools that follow the program of a special high school for gifted students – the Mathematical Grammar School. All other contestants are included in division B. The main difference between the problems in these two divisions is their difficulty. Furthermore, the minimum number of points needed for advancing to the next stage can be different between divisions.

For the Regional and National competitions in Serbia, there is no global training. However, there are some online tournaments via the online judge system (Petlja, 2018). Furthermore, some special schools organize preparations, which vary from town to town. A training camp for the Serbian national team (for JBOI/EJOI or BOI/IOI) is organized every year. Most of the times, the training camp is organized in Belgrade or some mountain camp (Divcibare, Petnica), where students have lectures all day. The training is structured so that there are two basic modes of practice: 2 hours of theoretical lecture (on a blackboard) in the morning, and 4–5 hours of coding in the evening. Typically, this training lasts for 5 days.

As Hadzi-Puric explains, “Before 2016, we didn’t have an official support for training camps. Our society, Mathematical Society of Serbia (DMS, 1948), had received a portion of the money from the Ministry of Education, but just to cover travel costs for an international Olympiad. In the last two years, the Ministry of Trade, Tourism, and Telecommunications organized a public invitation to all interested associations and foundations, having the status of non-government organizations, to submit program proposals for co-financing from the funds allocated from the budget of the Republic of Serbia. The subject of the invitation was granting funds (10 000 euros) for programs in the field of information society development, i.e. within the international competitions in the field of computer science, mathematics and physics.”

According to Ranko Cabrilo, assistant director for IT at the Examination Centre of Montenegro (ECM, 2005), the annual computer programming competitions in Montenegro are organized in two levels. These are the School level competition and the State competition. Condition to qualify for the State competition is to win more than 50% of the total points in the School competition. Since in Montenegro, computer programming is an elective and not a regular subject in high schools (where most students come from), they don’t have a huge number of candidates. Prior experiences state that the number of candidates is usually between 15 and 20. Most of the candidates successfully qualify for the State competition. The school competition is organized and financed by schools. All state competitions (including computer programming), from which students qualify for the Balkan and the International Olympiads, are organized by ECM. After the State competition, ECM organizes trainings for the best competitors, and it also covers the travel expenses for international competitions.

As Cabrilo explains, “it is difficult to estimate the budget for covering all the expenses regarding computer programming, as all competitions in various subjects are

also financed as computer programming. These subjects include Mathematics, Physics, Chemistry, Biology, Geography, etc." In any case, if there is shortage of assets, ECM tries to include different sponsors and donors in order to smoothly organize all competitions and travels.

The competition cycle in **Slovenia**, as Simon Weiss – the current deputy leader of the IOI team explains, includes: 1) School level competition; 2) State level competition; and 3) Playoff competition for IOI/BOI team selection. The School level competition is conducted in schools. The best contestants from this competition are invited to participate in the State competition. For the State competition, contestants are divided into 3 groups: easy group (implementation and basic algorithms), medium group (which requires many different skills from contestants), and advanced group (for the best contestants). All the contestants that achieve a reasonable score from the advanced group, the best few contestants from the medium group, and the best contestant from the easy group, are then invited to participate in a one-day Playoff competition for selection of teams for BOI/IOI. Typically, the number of participants in the School level competition is over 200, 75% of which qualify for the State competition, and 15 contestants make it to the Playoff competition.

As elaborated by Constantin Galatan, the national coordinator for the contests in informatics, in **Romania**, the NOI and IOI related activities are directed and financed by the Ministry of Education of Romania. The National Olympiad in Informatics (abbreviated as ONI in Romanian) has two sections. The first section includes lower-secondary school students: 5th to 8th grade (10–14 years of age). The second section is for high school students: 9th to 12th grade (15–18 years of age). Similarly, the National Training Team consists of two distinct sections: the Junior Training Team, addressed to students from lower-secondary school level (10–14 years of age); and the Senior Training Team, composed of upper-secondary school level students (15–18 years of age).

The Romanian National Olympiad in Informatics is organized in three stages: 1) Municipal Olympiad in Informatics (OMI); 2) County Olympiad in Informatics (OJI); 3) National Olympiad in Informatics (ONI). The National Training Team Selection includes another three stages: 4) Selection contest for the Junior and Senior National Training Teams; 5) First training camp; 6) Second training camp.

For the first three stages, the students from 5th to 10th grade receive different tasks according to the level of study. The tasks for the 11th and 12th grades are the same. Once ONI is finished, the National Training Team is selected in the following manner: first, the Selection Contest for the Junior and Senior National Training Teams is organized and it is a one-day onsite contest. The first 50% of the students in the ONI contest in each level of lower-secondary school take part in this Selection contest. The same happens in the case of high-school students who participate in the Selection contest for the Senior National Training Team. In order to select the National Team, the results from ONI are also taken into consideration.

Regarding the training of the IOI candidates in Romania, two dedicated training camps are organized, 7 days each. Usually, the first training camp takes place at the end of April or early May, and the second one in mid-May. In between, there are online ses-

sions, but they are not part of the official training schedule of students preparing for IOI. They participate in contests organized by Romanian websites such as infoarena.ro.

According to Biserka Yovcheva, professor involved in the Bulgarian competitions, the **Bulgarian** national team for the international competitions is selected in several stages. First, the broader national team is selected through the National Olympiad in Informatics (NOI). NOI is organized in three stages:

- (1) Municipal competition – organized by local teachers in schools, on town level. Typically, more than 1000 students participate in the competition.
- (2) Regional competition – also conducted onsite, on town level, but using tasks prepared by the National Committee of the Ministry of Education and Science. The task solutions are checked by the National Committee and around 120 contestants qualify for the next stage, in 5 groups: E, D, C, B, and A.
- (3) National competition – the final competition. Typically, the best 25 contestants from the group C, 20 contestants from the group B, 32 contestants from the group A, and 40 contestants from groups D and E gain the right to participate in this competition.

The junior national team is selected from the participants of the group C, while the senior national team is selected from the participants of the groups A and B. The best 12 junior contestants and the best 12 senior contestants from the National competitions are invited to take part in additional 4–5 competitions before the national teams of 4 students (each) are selected.

Preparations for the selected teams (senior/junior) are conducted twice per year – a 3 day preparation camp in June, in Sofia, and another 7-day camp in July. So far, these preparations have been financed by the “America for Bulgaria” foundation.

There is a project within the Ministry of Education and Science in Bulgaria, named “Student Olympiads and competitions”, for covering the finances for all the activities in the annual competition cycle. However, the project doesn’t always cover the costs completely, so sometimes schools and sponsors provide some of the money. Furthermore, there is a Natural Science Olympic Teams Society, which is funded by the “America for Bulgaria” foundation, and which also participates in the financing.

As described in (Can *et al.*, 2015), the **Turkish** national team for IOI/BOI is selected after the conduction of 3 contests and 2 training camps. First, a paper-based exam on national level is held to select students that will participate in a summer camp. Nearly 1300 students from all over Turkey participate in this first-level exam, which usually is held in May. The top 55 students from the exam qualify for the summer camp. The summer camp lasts for 2 weeks, starting in late August and ending in early September. Lecturers in this camp are typically academics from the most respected universities in Turkey. After the summer camp, another contest is organized, and students that achieve high scores in this contest are invited to participate in a winter camp. The contest is performed in two days, according to the IOI standards, and it takes place in November. The best 18 students in this second contest are allowed to attend the winter camp. The winter camp also lasts for 2 weeks, during February. The final step of the selection process is yet another contest following the winter camp, from which the best 4 students are selected to be part of the national team that represents Turkey at the international competitions – IOI and BOI.

Preparations for the national IOI/BOI team are organized in the 2 weeks preceding the actual IOI. These preparations are usually held in Ankara, and include a contest in each of the preparation days.

A critical stakeholder, which provides the finances and maintains the organizational structure of the Olympiads in Turkey, not just in Informatics, but also in other subjects (such as Mathematics, Physics, Chemistry, and Biology), is the government agency TUBITAK. Each year, TUBITAK forms an official scientific executive committee, consisting of 3 academics from universities, and this committee is responsible for all the scientific matters – from task preparation to selection and training of national teams. The scientific committee, in collaboration with TUBITAK, organizes the paper-based exam in the first level as well as the training camps, and executes all the administrative tasks regarding the participation of the national team in IOI. It also recruits faculty members, interested graduate students, as well as alumni from previous years, to teach the contestants and help them practice for competitions during the training camps.

Competitions in informatics are held in **Macedonia** since 1990, and, by 2018, 29 national contest cycles have been conducted – which include multiple competitions each year, selection contests for international competitions, as well as training camps. Every year the contestants go through many levels of competition so that the best could be selected. The selected pupils represent themselves and Macedonia at the (Junior) Balkan Olympiad in Informatics (BOI/JBOI), European Junior Olympiad in Informatics (EJOI) and the International Olympiad in Informatics (IOI), as well as at other smaller regional competitions. The main organizer of the competitions in informatics for primary and secondary school pupils is the Computer Society of Macedonia (Kostadinov *et al.*, 2015). The total budget for the complete cycle that is raised mainly by sponsors is around 8000 euros per year.

The format of the competitions evolves each year, depending on many factors, such as the number of interested pupils, available resources, inclusion of programming in the schools' curricula, etc. The number of participants in the base contest is approx. 450, including around 50 juniors. Presently, the competitions are organized for primary and secondary school pupils, and include: School Qualification Competition, Regional/Municipality Competition, National Competition, National Olympiad, and (potentially) Selection Contests for International Competitions.

In order to support several competition types, to enable a large number of students to participate in the competitions, and to introduce as many pupils as possible to the art of programming, all of the competitions in informatics that are part of the national contest cycle (accredited by the Ministry of Education), are organized using the MENDO competition management system (Kostadinov *et al.*, 2010).

We have summarized the gathered information above in Table 1. As it can be seen, different countries include different **number of students in the first (base) contest, which is not strictly proportional to the population** of the respective countries. The smallest number of students is in Montenegro (only 15 to 20), and the largest number is in Romania (4000). Turkey involves 1300 pupils in the competitions although it is the country with highest population among the examined countries. The **number of cycles** through which the teams for international contests are elected also **varies**

from 2 (Montenegro) **to 5** (Romania, Bulgaria, Macedonia). Training of contestants is present in each country, but the form, quantity and regularity differs. **Finances**, as an important issue, vary from country to country. Firstly, some of the countries have organized state institutional support, while others depend on sponsors and sporadic institutional support. **The budget varies from 7 000 to 80 000 euros** (as an approximative value, since the support is given from different state institutions for different activities throughout the annual cycle).

Table 1
Summary of current situations in selected countries from SEE

Country	Levels in the competition cycle	Number of 1 st level participants	Training of contestants	Financing
Serbia	Municipal, Regional, National, Serbian Olympiad in Informatics (SIO)	Over 1200 senior and around 750 junior contestants	Online tournaments (via the online judge system Petlja),	Before 2016, no financial support whatsoever for training camps, only travel costs for international Olympiads covered by Ministry of Education. Starting from 2016, a budget of 10 000 euros is allocated for training camps by the Ministry of Education, Tourism and Telecommunications of Republic of Serbia (JBOI/EJOI and BOI/IOI)
Montenegro	School level competition, State competition	Between 15 and 20	Training camp for the best contestants in the State competition	School competitions are organized for the best contestants in the competitions and training camp organized by the Examination Center of Montenegro, which also covers the travel costs for international Olympiads
Slovenia	School level competition, State competition, Playoff competition for IOI/BOI team selection	120 (plus 100 junior contestants)	Preparations (2 lecture cycles) for the best contestants in the State competition	Budget: 7 000 euros
Romania	Municipal Olympiad in Informatics (OMI), Country Olympiad in Informatics (OJI), National Olympiad in Informatics (ONI), Selection Contest for Junior and Senior National Training Team, First and Second training camp	Around 4000 students	Two training camps (7 days each), online contests organized by Romanian websites such as infoarena.ro	All the activities are directed and financed by the Ministry of Education of Romania

Continued on next page

Table 1 – continued from previous page

Country	Levels in the competition cycle	Number of 1 st level participants	Training of contestants	Financing
Bulgaria	Municipal, Regional, National Olympiad in Informatics, 4-5 additional competitions for the national (IOI/BOI/JBOI/EJOI) team selection	More than 1000 students	Preparations for the selected teams for IOI/ BOI – 2 camps (a 3-days and a 7-days camp)	Budget: 80 000 euros. There is a national project for financing by the Ministry of Education, but sometimes schools, sponsors or even parents participate financially. There are also other sources of financing as well, e.g. the “America for Bulgaria” foundation
Turkey	First National Paper-Based Exam, Programming Contest, Team Selection Contest (IOI/BOI)	Approximately 1300 students	Summer School (2 weeks), Winter School (2 weeks), IOI camp (2 weeks)	Financing provided by the government agency TUBITAK. Budget: 40 000 US dollars
Macedonia	School Qualification Competition, Regional and National Competition, National Olympiad, and (potentially) Selection Contests for International Competitions	Approximately 450 students	Sporadic summer schools (around 1 week)	Financing provided by sponsors of Computer Society of Macedonia. Budget: 8 000 euros

4. A New Approach for Comparison of the Countries' Achievements

Each country is represented at the international Olympiads with a fixed number of candidates, i.e. each country may bring up to 4 contestants at IOI, up to 6 contestants at IMO, and so on. This clearly creates a challenge if someone wants to measure the achievement of one country compared to another. The main inequality is that each country is represented with a fixed number of contestants instead of some kind of proportional or merit-based quotas. A country with population of two million people is obviously expected to achieve a lower result than a country with population of 20 million, or even two billion. The question is – by how much?

As stated in (Taylor, 2012): “The attitudes of the IMO and IOI are almost diametrically opposed when it comes to publication of results. Although earlier I recall the attitude of the IMO to scores was to emphasize that the IMO was an individual rather than team event, IMO now on its official web site <https://www.imo-official.org/results.aspx> publishes the results of every student question by question, even if they get no points, and they also provide official placing for each country each year. This is accepted and completely non-controversial within the IMO community. There are two possible ways of providing a premiership for countries. One would be **to use every point won by the student** (as IMO does), while the other, which I have seen done by Australian

colleagues from other science Olympiads, is ***to use a medal count based on something like 3, 2 and 1 points for Gold, Silver and Bronze.”***

In the last years IOI also started publishing the complete lists of results for each student, so the principle mentioned above may also be applied for these results as well.

Anyhow, these approaches that we see at the Olympiads' websites and the websites related to people from Olympiad communities do not consider the characteristics of each country, especially the country's population.

Our approach proposal is to take into account the population of the countries in question. For example, if you consider country A with population of a and country B with population of b , then one should expect that out of the first n students in the ranking of the students from both countries, approx. $C_a = (a * n) / (a + b)$ should be from country A and approx. $C_b = (b * n) / (a + b)$ from country B, and this should stand for every n , when the countries are equally strong.

For example, if A is a country with population of 2 000 000, and B is a country with population of 7 000 000, then for $n = 2$, $C_a = 0.44$ and $C_b = 1.55$, which means that from the first 2 students either both should be from country B or in rare occasions one from each country. For $n = 4$, $C_a = 0.88$ and $C_b = 3.11$, which means that from the first 4 students either one should be from country A and 3 from country B or in very rare occasions all of them should be from country B. For $n = 6$, $C_a = 1.33$ and $C_b = 4.66$, which means that from the first 6 students either one or rarely 2 should be from country A and 5 or rarely 4 from country B.

From the above example, and if we consider that at IOI every country participates with 4 contestants, for the particular 2 countries (A and B), if the country A has 1 student ranked in the first 4 or 5 students from the joined ranking list, then one may consider that these 2 countries have achieved similar results. Every situation in which a student from country A is ranked higher than fourth place in the joined ranking gives “advantage” to the country A.

We believe that this approach is much more fair and precise, when used to compare 2 countries. Although it cannot be efficiently used when the population of the countries is dramatically different (i.e. the ratio B/A or A/B is greater than 5) for a specific year, it becomes usable when comparing the countries over the period of more than one (consecutive) years, by putting the contestants from all years in a joined ranking list.

This comparison approach that we propose may be used for analysis of the achievements of a country. The result may confirm (or throw away) the methods and approaches used in that country in their education and competition cycles organization.

In the following section we will give a short case study with focus on the results achieved by Macedonian contestants, compared to some of the neighboring countries.

5. A Case Study of the New Approach for the Macedonian Achievements

Organizer of the informatics competitions in Macedonia is the Computer Society of Macedonia (CSM). The organization of state events deals with large number of participants, which are mostly minors (under 18 years old) and, also, with the necessary

technology on site that requires staff to deal with technology malfunctions. CSM is a nongovernmental, non-profit organization and bases the organization of the competitions solely on sponsorships from companies, educational institutions, and sporadically from donations based on application in some calls for projects. With low finances in mind, CSM employs as cost-effective as possible ways for engaging pupils in the competitions, motivating teachers and school authorities, as well as keeping the participants informed, and “in good condition”. Among the challenges that CSM faced in the beginning of the last decade was the small number of children involved in the competitions, which, of course, led to results that were not noticeable in the worldwide IOI ranking (Jovanov *et al.*, 2017).

The challenges mentioned above inspired continuous evolution of the format of the competitions. The changes undertaken ever since depend on many factors, such as the number of interested pupils, the inclusion of programming in the schools' curricula, etc. Also, in the year 2010 the competition system called MENDO was developed and introduced in the management and execution of the competitions (Kostadinov *et al.*, 2010). All the improvements throughout the years, led to enhancement of the interest of pupils, and more and more pupils joined the competition process. From 48 pupils involved in the first round of competitions in year 2009, the interest rose to approx. 450 pupils in 2017. All of the aforementioned improvements, in turn, resulted in higher quality in the pupils' achievements. The achievements of the pupils in the Olympiads in Informatics are presented in Fig. 1. In this figure, the medals won by Macedonian competitors at the international competitions in informatics (IOI, BOI, and JBOI), in the period 1996–2016 are shown. There is a very obvious increase in the number of medals won, prior and following 2010. The last period is the one that involves the

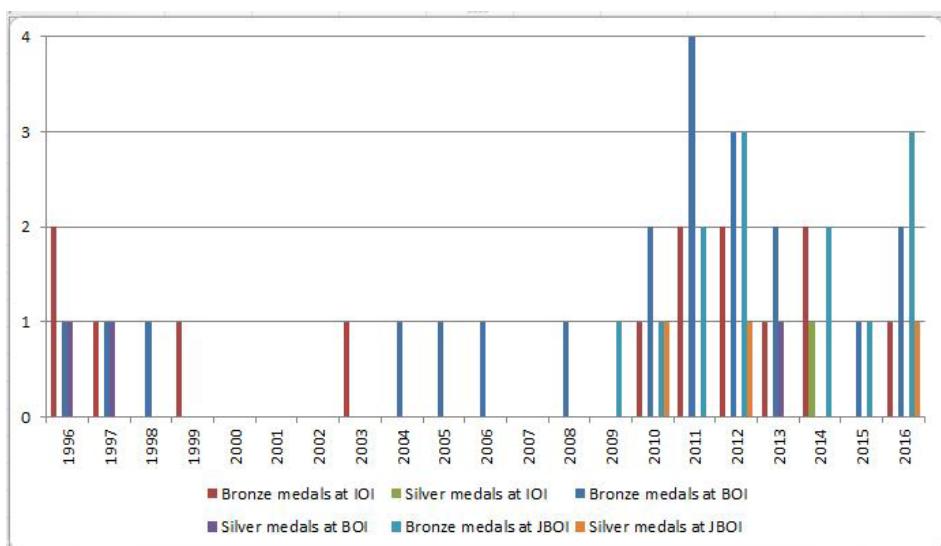


Fig. 1. Medals won by Macedonian competitors at the international competitions in informatics (IOI, BOI, and JBOI), in the period 1996–2016 (Jovanov *et al.*, 2017).

great changes in the competitions (the form and the tools for organization) and in the educational curriculum (Jovanov *et al.*, 2017).

Anyhow, as organizers of the competitions, we always ask ourselves the following questions: Have we managed to reach the maximum result for Macedonia? Is Macedonia's performance on the same level as the neighboring countries, even without the regular support from the government and the state budget?

The population of Macedonia is around 2 000 000 (CIA, 2018). The highest achieving neighboring countries of Macedonia (countries from SEE region) are Serbia (population around 7 000 000 (SORS, 2018)), Bulgaria (population around 7 000 000 (NSI, 2017)), and Romania (population around 20 000 000 (INS, 2012)). If we employ the new approach, it would mean that in the competition including the best participants from Macedonia and Bulgaria or Serbia, there should be one Macedonian in the first 4 students (based on the calculations from the previous section), or in a match between Macedonia and Romania, in the first 11 students there should be one Macedonian. Since IOI is usually attended by the four best contestants from each country, whenever one contestant of Macedonia is placed in front of a student from Bulgaria and Serbia, this is a success equal to the success of those two countries. In the last two IOIs the best Macedonian competitor has always been ranked in front of at least one competitor from Bulgaria, Serbia and Romania. Therefore, we may consider the Macedonian results at IOI at least equal to the strongest countries in our region.

We would not go into further deliberations and comparisons, because the focus of this paper is on the applicability of the new approach, and not on the particular achievements of Macedonia. ***Presented case clearly shows its applicability.***

6. Conclusion

In this paper, at the beginning, we have identified and summarized the necessary topics important for achieving good results at international scientific Olympiads. Further, we have presented thorough analysis of the contest systems of selected countries from South Eastern Europe (SEE) in the field of Informatics (Computer Science), as a region that is one of the prominent world regions in the context of high results in the international competitions. The characteristics of the informatics contest systems of Romania, Bulgaria, Serbia, Montenegro, Slovenia, Macedonia and Turkey were then summarized in a table, for easy comparison.

Further we presented a new approach that may be used to compare the achievements of countries based on the results that students achieved at Olympiads, and then we gave an application of this approach on the results of some of the discussed countries, compared to Macedonia.

We strongly believe that the content presented here, and the given approach for comparison, will be a valuable tool for the entities involved in the organization of the contests, to measure their results compared to other countries, to use the information for improvement, and to use their achievements to raise awareness among the government institutions and companies in order to get support from them.

Acknowledgement

The research presented in this paper is partly supported by the Faculty of Computer Science and Engineering, at the Ss. Cyril and Methodius University in Skopje. Authors wish to thank Jelena Hadzi-Puric, Ranko Cabrilo, Simon Weiss, Constantin Galatan, Biserka Yovcheva and Fatih Demirci, for supplying the information regarding the current situation with the national competition cycle in their respective countries.

References

- CAN, T., SIĞIRCI, İ.O., ABUL, O., DEMİRCİ, M.F. (2015). Informatics Olympiads in Turkey: team selection and training. *Olympiads in Informatics*, 9, 225–232.
- CIA – Central Intelligence Agency (1947–2018): the world factbook. (Accessed 19/5/2018).
<https://www.cia.gov/library/publications/the-world-factbook/geos/mk.html>
- Cormen, T.H., Leiserson, C.E., Rivest R.L., Stein C. (2009). *Introduction to Algorithms*, Third Edition.
- Dagienė, V., Mannila, L., Poranen, T., Rolandsson, L., Stupurienė, G. (2014). Reasoning on children's cognitive skills in an informatics contest: findings and discoveries from Finland, Lithuania, and Sweden. In: Gülbahar Y., Karataş E. (Eds) *Informatics in Schools. Teaching and Learning Perspectives. ISSEP 2014*. Lecture Notes in Computer Science, vol. 8730. Springer, Cham.
- DMS – Mathematical Society of Serbia (1948–2018).
<https://dms.rs/> (accessed 19/5/2018).
- ECM – Examination Centre of Montenegro (2005–2018). (Accessed 19/5/2018).
<http://www.iccg.co.me/1/index.php?lang=en>
- Guerra, V., Kuhnt, B., Blöchliger, I. (2012). Informatics at school-worldwide. In: *An international. Tech. rep.* Universität Zürich.
- INS – Institutul Național de Statistică (2012). Rezultate definitive ale Recensământului Populației și al Locuințelor – 2011 (caracteristici demografice ale populației).
- IOI – International Olympiad in Informatics (1989–2018).
<http://www.ioinformatics.org> (accessed 19/5/2018)
- Jovanov, M., Stankov, E., Mihova, M., Ristov, S., Gusev, M. (2016). Computing as a new compulsory subject in the Macedonian primary schools curriculum. In: *2016 IEEE Global Engineering Education Conference (EDUCON)*, 680–685. IEEE.
- Jovanov, M., Ackovska, N., Stankov, E., Mihova, M., Gusev, M. (2017). A decade of engineering computer engineers. In: *2017 IEEE Global Engineering Education Conference (EDUCON)*. Athens, 1309–1315.
- 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., Jovanov, M., Stankov, E., Mihova, M., Stojkoska Risteska, B. (2015). Different approaches for making the initial selection of talented students in programming competitions. *Olympiads in Informatics*, 9, 113–125.
- NSI – National Statistical Institute of Bulgaria (1880–2018). (Accessed 13/3/2017).
<http://www.nsi.bg>
- Petlja – Online judge system in Serbia. (Accessed 19/5/2018).
<http://petlja.org>
- SORS – Statistical office of the Republic of Serbia (1862–2018). (Accessed 19/5/2018). stat.gov.rs
- Taylor, P. (2012). Comparisons of the IMO and IOI. *Olympiads in Informatics*, 6, 199–204.
- Verhoeff, T. (1997). The Role of Competitions in Education. *Future World: Educating for the 21st Century: a conference and exhibition at IOI*.
- Verhoeff, T. (2011). Beyond the Competitive Aspect of the IOI: It Is All about Caring for Talent. *Olympiads in Informatics*, 5, 120–127.



M. Jovanov is an assistant professor at the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, in Skopje. As the President of the Computer Society of Macedonia, he has actively participated in the organization and realization of the Macedonian national competitions and Olympiads in informatics since 2001. He has been a team leader for the Macedonian team at International Olympiads in Informatics since 2006. His research interests include development of new algorithms, future web, and e-education.



M. Mihova is a professor at the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, in Skopje. She is a member of the board of the Computer Society of Macedonia. Her research interest is in the field of applied mathematics, more specifically applied probability and statistics, with focus on mathematical models in reliability, especially reliability of multi-state systems.



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 one of the organizers of the national competitions in informatics in Macedonia, and the Beaver event.



E. Stankov is a teaching and research assistant at the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, in Skopje. He is a member of the Executive Board of the Computer Society of Macedonia, and has actively participated in the organization and realization of the Macedonian national competitions and Olympiads in informatics since 2009. Currently he is a Ph.D. student at the Faculty of Computer Science and Engineering. His research includes analysis of program code correctness using different techniques, and its application to e-learning.

National Survey of Japanese Universities on Computing Education: Analysis of Departments Majored in Computing Discipline

Tetsuro KAKESHITA

Department of Information Science, Saga University

Honjo 1, Saga, Japan

e-mail: kake@is.saga-u.ac.jp

Abstract. We conducted the first national survey of computing education at Japanese universities in 2016. In this paper, we report the survey result of the computing education at a department or a course majored in the computing discipline. The survey covers various aspects including program organization, quality and quantity of educational achievement, students, teaching staff and computing environment. Thus the survey result is expected to be a good fundamental to develop realistic computing curricula and accreditation criteria in Japan. The estimated number of computing departments and students in Japan is about 300 and 28,000 respectively. 50% of the students belong to engineering faculties. Although 25% of the students are learning Computer Science, 50% of the students are learning computing domains other than those defined in CC2005. The information processing society of Japan (IPSJ) and the Japanese Ministry of Education (MEXT) utilize the survey result to develop a new computing curriculum standard J17 and national policy of computing education respectively.

Keywords: web-based survey and analysis, college level education, curriculum development and analysis, accreditation criteria development, computing education, **quality assurance in education.**

1. Introduction

Computing education is essential at modern universities since information technology is expected as a powerful innovation driver as well as an essential infrastructure of the modern society. There are four types of computing education in Japanese universities.

1. Computing education at a department or a course majored in computing discipline.
2. Computing education at a non-IT department or a course as a part of their major field of study.

3. General computing education for all university students typically at the first or second academic year.
4. Computing education to obtain high school teacher license on computing subjects.

We conducted a national survey of Japanese universities on computing education in 2016 (Kakeshita, 2017). The survey is composed of five survey types A through D described above as well as the survey type E for educational computer system. Our survey is actually the first national survey in Japan since there was no widely accepted definition of computing education. This situation is essentially the same at other countries so that we are not aware of a similar survey collecting comparable level of detailed data as our survey.

The Science Council of Japan developed the reference standard of informatics (Hagiya, 2015) for university education in 2016. The reference standard provides a common body of knowledge (BOK) for college level computing education and the Japanese government accepted this as a definition of computing education. Thus we shall use the reference standard as the definition of computing education in this paper.

In this paper, we report and discuss the result of the survey type A for computing education at a department or a course majored in computing discipline. The survey is designed to analyze and understand current status of computing education at Japanese universities from various aspects including program organization, quality and quantity of educational achievement, students, teaching staff and computing environment. The analysis result will be expected as a fundamental to develop reasonable curriculum guidelines and accreditation criteria for computing education. The analysis also clarifies the difference of the five major computing domains, CS, CE, SE, IS and IT, which are developed separately by different community.

The Information Processing Society of Japan (IPSJ) utilized the result to develop the new J17 curriculum standard for computing education in 2017. The Japanese Ministry of Education (MEXT) will utilize the survey result to improve the national policy of computing education.

2. Survey Plan

2.1. Survey Questions

The following is the list of questions for survey type A. As the reader can understand from the list, our survey covers various aspects of computing education. These questions are prepared by considering the Japanese standards for establishment of universities and our accreditation experience of computing programs in Japan.

- Name of university, faculty, department and course.
- Program organization.
 - Day time, night or remote program.

- Academic discipline of the program such as engineering, social science and humanities.
- Specific computing domain including CS, CE, SE, IS and IT defined in CC2005 (JTF, 2005).
- Required number of credits for graduation.
 - Number of subjects provided.
- Quality and quantity of educational achievement.
 - See Section 2.2 for detail.
- Enrolled students.
 - Regular academic years of the program.
 - Number of students.
 - Student's choice of career after graduation.
- Teaching Staff.
 - Number, educational background, current specialized field, tenure of faculty members.
 - Number and workload of support staff.
 - Number and workload of teaching assistant students.
- Computing environment.
 - Educational computer system.
 - Utilization of Student's own PC.
 - Educational programming language.
- Other topics (If any).
 - Future plan and strength of the program.
 - Utilization of IT certification and/or qualification.
 - Special remarks.

2.2. Survey of Quality and Quantity of Educational Achievements

The survey of quality and quantity of educational achievements is the core of our survey. We define six achievement levels for knowledge and skill defined in Table 1. These levels are used to define quality of education.

We also define a BOK based on the reference standard of informatics (Hagiya, 2015) and additional topics related to general computing education (Kawamura, 2008). The BOK contains 90 topics classified by 21 domains as represented in Table 2. The BOK is used to precisely define educational contents of each program.

Although CC2005 (JTF, 2005) defines five computing domains to define typical computing curriculums, corresponding BOKs are different depending on the domain. We define the BOK instead of using the five different BOKs. We also expect to find educational programs other than the typical ones. Common BOK is also useful to clarify the difference among the existing five domains (Kakeshita and Ohtsuki, 2014).

It is usual that a computing education as the major domain is performed at a department or a course of a department. Thus a department or a course responds to the survey. Each organization answers expected knowledge and skill levels of the students at each

Table 1
Knowledge and Skill Level Definition

Level	Knowledge Level Definition	Skill Level Definition
0	Not taught (unnecessary or already taught at general computing education)	
1	Not taught because of the time limitation or because the level of the contents is too high	Taught at class with simple exercise
2	Taught at class. Students know each term	Taught at class with some exercise. Students can perform the topic if detailed instruction is provided
3	Taught at class. Students can explain the meaning of each term	Taught at experiment with more complex exercise. Students can perform the topic with simplified instruction
4	Taught at class. Students can explain relationship and/or difference among related terms	Students perform combined research project containing the topic so that the students can autonomously perform the topic
5	Taught at class or graduation research project. Students can teach related domain or subject of the terms to the others	Students perform combined research theme containing the topic. Students can teach how to perform the topic to others

Table 2
Common BOK Organization

Source	Section	Domain	# of Topics
J07-GEBOK	General Computing Education		9
Reference Standard for Informatics	(A) General Principles of Information		6
	(B) Principles of Information Processing by Computers	Information Transformation and Transmission Information Representation, Accumulation and Management Information Recognition and Analysis Computation Algorithms	4 4 4 6 8
	(C) Technologies for constructing computers that process information	Computer Hardware I/O Device Fundamental Software	3 4 3
	(D) Understanding humans and societies that process information	Process and Mechanism for Information Creation and Transmission Human Characteristics and Social System Economic System and Information IT-based Culture Transition from Modern Society to Post Modern Society	2 3 2 2 2
	(E) Technologies and organizations for constructing and operating systems that process information in societies	Technics for Information System Development Technics for Information System Utilization Social System Related to Information Principle and Design Methodology for HCI	7 6 2 4
	Professional Competency for IT Students		3
	Generic Skill for IT Students		6

topic of the BOK. At the same time, the organization answers the total number of students taking a subject teaching each topic. As a result, quality and quantity of education at the organization can be summarized through the survey.

2.3. Survey Process

We prepared the survey in October 2016. At first we defined the survey questions and set up the web-based survey system (Kakeshita, 2011). We utilized the web-based survey since we did not exactly know the actual organizations for this survey in advance. After preparing various document such as user manual and detailed instruction of the survey questions, we sent the formal request letter to all universities in Japan with a reference letter from the Japanese Ministry of Education in order to increase the response rate.

The survey was executed for two months starting at the beginning of November 2016. Each survey responder of survey type A must first register to the web system and then answer the questions listed in Section 2.1. We also provided FAQ at the survey web site and sent e-mail responses to each of the questions from the survey responders.

After closing the survey, we reviewed the collected answers and requested the responders for possible correction of the incomplete answers.

3. Overview of the Survey Result

3.1. Response Rate Analysis

We collected 296 answers for the survey type A. We reviewed the answers and found that 17 are invalid because of the two reasons. (1) The number of required credits is less than 30, which is 25% of the minimum credits defined by the Japanese standards for establishment of universities. (2) The name of the department does not indicate computing discipline. After contacting the 17 organizations, we obtained permission to exclude them from the computing departments. Therefore the total number of answers is 279.

Among the 279 answers, 82 come from national universities and 166 from private universities. 31 answers come from public universities which are founded and run by local government such as city or prefecture.

Each answer is provided either by a faculty, collection of departments, single department or a course. Thus the number of answers does not directly represent the actual number of computing departments in Japan. We examined the answers to have Table 3 representing the number of universities, faculties and departments having computing department or course.

Table 3
Number of Computing Departments in Japan

	University	Faculty	Department
National	53	61	75
Public	22	22	29
Private	108	133	163
Total	183	216	267

There is a council of informatics departments (DI-Council) in Japan whose academic discipline is natural science or engineering. 151 departments join the DI-Council. 127 departments (84.1%) also respond to the survey so that we can estimate that the response rate of the survey is about 85%. Considering the response rate, we estimate that the number of computing departments in Japan is approximately 300. This means that there are quite a large number of computing departments which we are not aware in Japan.

The response rate 85% is quite high considering that each organization must independently register to the web-system. This becomes possible because of the strong support of the Ministry of Education, Japan. Each organization responds to the survey as a part of their job.

3.2. Student Enrollment

Table 4 represents the number of students majored in the computing discipline classified by academic discipline and specific domain within computing.

Total number of students is 26,112. Considering the response rate, the total number of computing students is approximately 28,000 in Japan. The numbers of male and female students are 21,529 and 4,583 respectively. 47.3% of the female students belong to 20 universities so that the distribution is highly skewed. Since IT service is widely

Table 4
Distribution of Computing Students Classified by Academic Discipline and Computing Domain

Academic Discipline	CS	CE	SE	IS	IT	Others	Total
Engineering	5,549	1,632	78	1,664	997	3,715	13,635
Social Science	40			419	742	3,453	4,654
Physical Science	730			100		224	1,054
Humanities					125	318	443
Pharmacy and Nursing				170	70	56	296
Art			65		78		143
Education						40	40
Others	449		157	20	665	4,556	5,847
Total	6,768	1,632	300	2,373	2,677	12,362	26,112

utilized in the modern society, it is expected that more female students study computing discipline as their major in order to promote further innovation using IT. The number of IT professionals is approximately 1 million in Japan so that more students are expected to learn computing discipline as their major.

According to US educational survey (NCES, 2017), the Bachelor's degree in computer and information science is 59,581 in 2014-15 so that the number of computing students in Japan is approximately 43.8% of that of US. Since the population of Japan is 39.0% of that of US, the ratio of computing students against the population is almost the same at both countries.

The following is the observations we found in Table 4:

- The second largest academic discipline is “Others”. The corresponding departments are interdisciplinary, i.e. belonging to two or more disciplines. This indicates the diversity nature of the computing discipline which cannot be covered by an existing discipline.
- 26.5% of the departments are teaching computer science (CS) and 32.3% are teaching one of the other existing computing domains. However, 41.2% of the departments cannot be covered by a single domain. Appropriate curriculum guidelines are expected to be developed for these departments.
- Although CS and CE are mainly taught at engineering or physical science departments, IS and IT are also taught at social science and other departments. We consider that there is a historical reason for this. Computing education, particularly CS and CE, has been provided mainly at a department majored in engineering or natural science. However the computing domain is expanding so that new domains such as IS and IT are emerging. The departments majored in social science and humanities also take the responsibility to provide IS and IT education.

Table 5 represents the computing student's career selection after graduation. The number of students does not coincide with Table 4 because of the incomplete answers. It should be noted that career selection is completely different at national, public and private universities. 52.8% of the students enter a graduate school at national university, while 9.9% at private university. A possible reason is that the tuition fee of a **private university** is typically two or three times higher than that of the national or public universities. It is also recognized that many of the students enter the graduate school of the same university they graduated. This means that 6-year education consists of undergraduate and master course education can be effectively introduced mainly at **national universities**. This also implies that mobility of the student is rather limited in Japan.

Table 5
Student's Career Selection after Graduation

Career Selection	National	Public	Private	Total
Graduate school in computing discipline	2,620	388	1,309	4,317
Graduate school in other disciplines	338	57	237	632
Hired at company, government or school	2,409	1,093	12,198	15,700
Others (including unknown)	231	52	1,828	2,111

3.3. Number of Credits for Graduation

In Japan, an undergraduate program must contain at least 124 credits to obtain a Bachelor degree. 1 credit requires 45 hours of learning including class. Typical curriculum is composed of general education at university level (typically 20 to 40 credits) and common education at faculty level (typically 10 to 20 credits) in addition to the specialized education at a department or a course. An educational institution must first assign credits to each component. Depth and width of the education is greatly affected by the allocated credits.

Fig. 1 represents the distribution of the required number of credits for the specialized computing education. The distribution is illustrated using a box plot and provides a realistic restriction to design computing curriculum for each domain. For example, typical CS curriculum is composed of 75 to 100 credits. Observing this distribution, it is recommended to design a CS curriculum guideline between 50 and 60 credits. This will allow freedom to design computing curriculum considering strength and background at each educational program while preserving quality assurance of computing education among many of the CS departments. Similar discussion is possible at other domains.

4. Educational Achievements

We shall analyze the educational achievement, i.e. quality and quantity of education, in this section. We collected 97 answers of the quality survey and 67 answers of the quantity survey. Although the number of collected answers is smaller than the number of responses to the survey, it is comparable to 75 which is the estimated number of samples calculated under the assumption of universe size 300 and 10% statistical error. Therefore our discussion can be statistically reasonable.

We define educational effort of a program for a certain topic of the BOK by the multiplication of average level value and the number of students learning the topic. We thus define two types of effort values to teach knowledge and skill.

Fig. 2 illustrates the effort distribution of each computing domain. Each of the effort values are summed up for each section as defined in Table 2 in order to clarify characteristics of each domain. Although skill effort values are used in Fig. 1, correlation

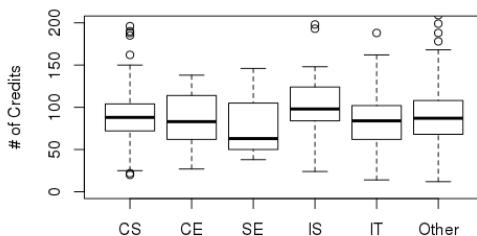


Fig. 1. Distribution of the Number of Credits for Graduation at Each Computing Domain.

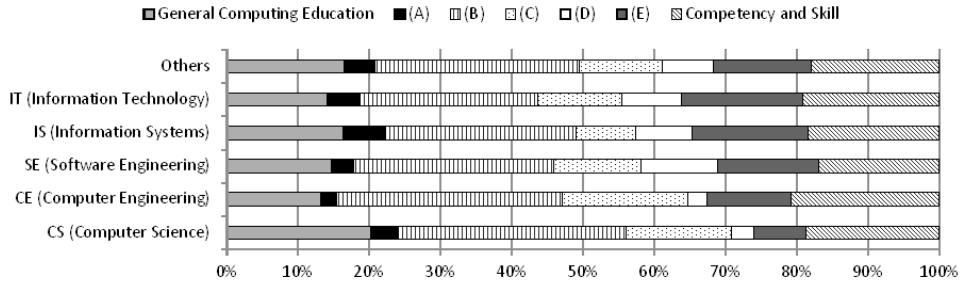


Fig. 2. Educational Effort Distribution of Skill Teaching for Each Computing Domain.

coefficient between skill and knowledge effort values is 0.97 so that distribution of the knowledge effort values is similar.

Major difference of the effort distribution among the domains arises at the sections from (B) to (E). In CS and CE, traditional contents defined in (B) and (C) are dominant. On the other hand, educational program of IS and IT focus more on (D) and (E). As we discussed in Section 3.2, there are some IS and IT departments in social science where (D) is contained in their major subjects. The contents in (E) can also be taught at social science departments because system operation is included in (E).

We also calculated the effort at each domain defined in Table 2. The top 3 domains with the highest effort are as follows. Many of the computing departments focus on these educational topics.

- General Computing Education
- Generic Skill for IT Students
- Algorithms

These domains are relatively easy compared to other domains. We consider that there are two reasons that the computing departments focus on these domains. One is the shortage of teaching staff who can teach high level computing topics. The other is the declining academic ability of the students due to the increase of the percentage of students who enter university or college in Japan.

On the other hand, the topics other than the above three can be considered as the contents which an educational institution can claim their originality. Since the computing discipline is quite large, each institution is expected to focus on particular domains which they have strength.

Fig. 3 summarizes the overall distribution of the skill levels of each computing domain. The skill level is summarized for each section of the BOK defined in Table 2. The skill level distribution at each section characterizes each computing domain. We can observe that typical skill level of general IT education is at most 2 so that students are taught with some exercise. However the typical skill level of competence exceeds 2. The difference comes from that competence is usually taught during the graduation thesis project while general IT education is taught at the first or second academic year. We also observe that the average knowledge level is around 3 so that we can expect that a typical student can explain the meaning of a technical term.

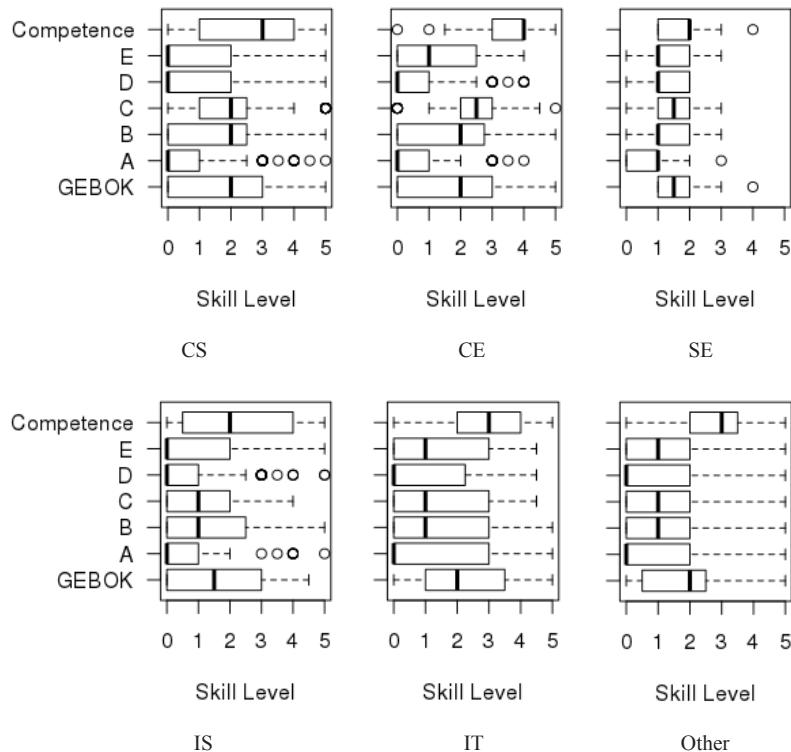
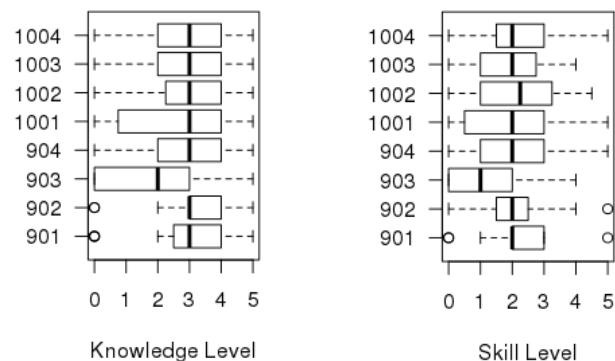


Fig. 3. Overall Distribution of Skill Levels of Each Computing Domain.



901: Data, 902: Data Structure, 903: Data Type, 904: Database, 1001: Signal Processing,
1002: Pattern Recognition, 1003: Machine Learning, 1004: Data Mining

Fig. 4. Detailed Distribution of Knowledge and Skill Levels of CS Domain (Selected Topics of Section B).

Fig. 4 represents the detailed distribution of knowledge and skill of the selected topics of section B for the CS domain. The readers can observe that the achievement levels of topic 903 (data type) is rather weak compared to other topics. Although we have similar distributions for the other combination of the other computing domain and topics, they are omitted due to the space limitation. The box plot provides top 25%, median (top 50%) and top 75% values of the levels so that the readers can consider desirable, typical and minimum levels for each section and/or topic.

These distributions are useful both for the computing departments and curriculum development. From the viewpoint of a computing department, the distribution is useful to analyze strength and weakness of their computing program for each topic. From the viewpoint of curriculum developer, the distribution can be utilized to define realistic requirements for the achievement levels at each topic. Such recommendation about the achievement level can also be utilized at computing accreditation.

5. Teaching Staff

5.1. Faculty Member

Table 7 represents the number of faculty members teaching at computing departments. Faculty members outside of the department are in charge of 23.2% of the classes. This indicates shortage of teachers at computing departments.

The number of organizations is 119 (49.6% of the valid answers) that the number of computing department graduates is less than 50% of the number of faculty members. The number of organizations is 68 (28.3%) that the number of computing department graduates is less than 30% of the faculty members. It is our concern that systematic computing education may not be enough for the graduates of non-computing departments.

We also observe that the ratio of the faculty members whose current major is computing is more than that of the ratio of computing department graduates. This means that a significant number of faculty members changed their major after graduation. However we also observe that the number of organizations is 59 (24.6%) that the ratio

Table 7
Number of Faculty Members at Computing Departments

	Total	Computing Dept. Graduates	Current Major is Computing	# of Classes in charge
Faculty members of the department (with tenure)	4,281	2,315 (54.1%)	2,943 (68.7%)	13,824
Faculty members of the department (without tenure)	643	341 (53.0%)	397 (61.7%)	1,459
Faculty members of other departments or faculties	1,200	459 (38.3%)	520 (43.3%)	2,461
Part-time instructor (outside of the university)	1,949	653 (33.5%)	940 (48.2%)	2,275

of faculty members is less than 50% whose current major is computing. Some of them answered that “the number of faculty members majored in computing is not enough” or “having help of the faculty members belonging to other departments”. We expect these organizations to have academic positions to hire more faculty members who have enough ability to teach contemporary computing technology.

5.2. Support Staff and Teaching Assistant

Although 78 organizations (67.5% of the valid answers) employ a support staff for computing education, 162 do not employ support staff. We also find that 49 organizations (20.4%) do not employ teaching assistant student. Major reason of these is considered as financial restrictions. Support staff and/or teaching assistant are essential in order to effectively support students during exercise and experiments to achieve expected skill levels. It is expected for the government and university to provide financial support.

The support staffs assist 541 classes with 1.6 staff per class. The teaching assistants assist 3,770 classes with 77.8 man-hours per class. Thus teaching assistant students are more important than support staffs at many classes. We found that some of the universities located in a metropolitan area employ students belonging to neighboring universities as their teaching assistant.

5.3. Student/Teacher Ratio

Fig. 5 represents the distributions of the number of students per teacher of the computing domains. The box plot represents top 25%, median (50%) and 75% values so that these values can be utilized to determine reasonable accreditation criteria (minimum requirement and recommendation). The student/teacher ratio is less than 10 at many of the organizations. However we found 23 organizations whose ratio exceeds 10. It is expected to keep appropriate number of students per teacher to provide enough care for the students.

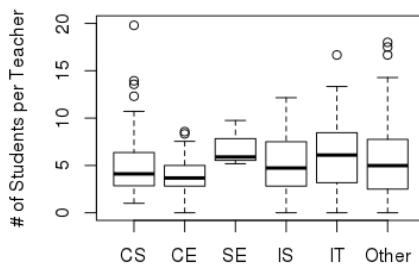


Fig. 5. Distribution of the Number of Students per Teacher of Each Domain.

6. Computing Environments

Table 8 represents situation of computing environments classified by educational computer system and student PC purchase. The item “does not use computer system” indicates that the department etc. does not use educational computer system provided by the university.

24.4% of the organizations have their own educational computer system so that they can fully control the system. However 23.6% do not have any computer system. 49.5% utilize shared computer system at various levels. This is because the computer system budget tends to be reduced at many universities. However, as demonstrated in Table 9, we find that utilization of computer system greatly affects average knowledge and skill levels of the students. The reader can observe from Table 9 that the average levels are similar between the institutions with private and shared educational computer systems. Although it is more difficult to control the shared computer system, some of the computing departments can effectively control the system in the case that they have faculty members majored in computer system administration.

Although BYOC concept is widely known, only 24.4% of the organizations require students to purchase their own PC. 67.4% leave students to decide whether to purchase PC. 52 organizations do not recommend students to purchase PC although they do not have educational computer system. We are planning to investigate the detailed reason as a future work.

Table 8
Educational Computing Environment at Computing Departments

	Required to Purchase Student PC (Dept.)	Student PC (Faculty)	Student PC (Univ.)	Student PC Recommended	Other	Total
Private Computer System	11	3		7	47	68
Shared Computer System (University Level)	2	5	14	2	48	71
Shared Computer System (Faculty Level)		10	1	7	21	39
Shared Computer System (Campus Level)	2	1	1	5	19	28
Does not use Computer System	3	3			1	7
No Computer System	3	1	7	3	52	66

Table 9
Relationship between Educational Computer system and
Average Knowledge and Skill Levels

Educational Computer System	Average Level	
	Knowledge	Skill
Private System	1.67	1.05
Shared System	1.67	1.08
No Computer System	1.20	0.73
Total	1.63	1.04

Table 10
Top 10 Educational Programming Languages

Programming Language	Score	Programming Language	Score
1. C	826.5	6. SQL	100.5
2. Java	602.5	7. Python	87.5
3. C++	205.0	8. Visual Basic/VBA	82.0
4. JavaScript	168.5	9. PHP	72.0
5. Assembly Language	117.0	10. Ruby	31.5

Table 10 represents the top 10 educational programming languages. The score is estimated by the weighted sum of the answers with priority. C and Java are the traditional choices as structured and object oriented language. C++ is utilized as “better C” because it supports simplified I/O (`cin/cout`) and STL (standard template library). Python and Ruby are utilized as simple languages with high software productivity.

The following is a list of comments from the organizations. Several difficulties can be found from the comments. It is expected to provide appropriate support for these organizations:

- We quitted mandatory programming courses so that its effect is under investigation.
- Some students do not understand importance of software engineering despite of complexity of software development.
- If a student perceives that he is not good at computer programming, the fact greatly affects student motivation.

7. Educational Efforts

Many organizations continuously improve their education program by curriculum updates or by faculty reorganization. Some of the typical educational activities collected through the survey are listed below:

- Introduction of good educational practices: PBL (problem/ project based learning), active learning, presentation of student research papers at academic societies, etc.
- Cooperation with industry or other universities.
- Obtain accreditation by JABEE (Japan Accreditation Board for Engineering Education).
- Adoption of new computing curricula such as CS2013 (JTF, 2013).

We also find activities related to certification or qualification in the computing discipline as listed below. Such IT qualifications will be useful for the students to get a job after graduation.

- Curriculum design considering typical IT qualification such as JITEE (IPA, 1969) which is a large qualification with 500,000 applicants each year provided by the Japanese government.
- Student support to obtain IT qualification: give credits to qualification holders, support seminar, funding support, etc.

8. Concluding Remarks

The analysis of the survey result will be a good input to design computing curriculum and accreditation criteria to improve computing education in Japan. Similar survey and analysis will be also valuable in other countries.

Information technologies and social situation are rapidly changing. Computing departments are receiving many educational requests from the industry and the society. Some of them are AI, big data, data science, IoT, innovation using IT and information security. We found that many computing departments are trying to respond to these requests through the survey.

We can observe the entire picture of the computing education at Japanese universities through the survey. Although several problems are discovered, IPSJ is willing to improve the current situation through development of new computing curriculum standard J17 and cooperation with Ministry of Education, Japan.

Acknowledgments

This survey project is supported by the Ministry of Education, Culture, Sports, Science and Technology, Japan. The authors greatly appreciate the faculty members and the administration officers of the universities who take time to respond to the survey. This research is also supported by JSPS KAKENHI Grant Numbers 16K01022 and 17K01036.

References

- Hagiya M. (2015). Defining informatics across Bun-kei and Ri-kei, *Journal of Information Processing*, 23(4), 525–530. DOI: <http://doi.org/10.2197/ipsjjip.23.525>
- IPA (1969). *Japan Information-Technology Engineers Examination*. Available at <https://www.jitec.ipa.go.jp/index-e.html>
- Joint Task Force (JTF) of ACM, AIS and IEEE-CS (2005), Computing Curricula 2005 (CC2005): The Overview Report. Available at <https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf>
- Joint Task Force (JTF) of ACM and IEEE-CS (2013). Computer Science Curricula 2013 (CS 2013). Available at https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf
- Kakeshita, T. (2011). A web-based survey system to analyze outcomes and requirements: a case for college level education and professional development in ICT. In: *Proc. 5-th Int. Conf. on Society, Cybernetics and Informatics (IMSCI 2011)*. 82–87.
- Kakeshita, T., Ohtsuki, M. (2014). Requirement analysis of computing curriculum standard J07 and Japan Information Technology Engineers Examination using ICT common body of knowledge. *Journal of Information Processing*, 22(1), 1–17. DOI: <http://doi.org/10.2197/ipsjjip.22.1>
- Kakeshita, T. (2017). National survey of Japanese universities on IT education: overview of the entire project and preliminary analysis. In: *Proc. 9-th Int. Conf. on Computer Supported Education (CSEDU 2017)*. Porto, Portugal, 607–618.
- Kawamura, K. (2008). Computing curriculum Standard J07: computing in general education (J07-GE). *IPSJ Magazine*, 49(7), 768–774. (In Japanese).
- National Center for Education Statistics (NCES). *Fast Facts*. Available at <https://nces.ed.gov/fastfacts/>



T. Kakeshita is an associate professor at Department of Information Science, Saga University, Japan. He received his Ph.D. degree in Computer Science from Kyushu University, Japan in 1989. His major research interests include quantitative analysis of ICT education and ICT certification, and complexity analysis of database and software systems. He received an excellent educator award from Information Processing Society of Japan (IPSJ) in 2013. He joined many activities such as IPSJ educational activity, Certified IT Professional Certificate (CITP), accreditation at Japan Accreditation Board for Engineering Education (JABEE) and ISO standard development (ISO/IEC JTC1/SC7/WG20).

Platform for Analysing and Encouraging Student Activity on Contest and E-learning Systems

Bojan KOSTADINOV, Mile JOVANOV, Emil STANKOV

Faculty of Computer Science and Engineering, University Ss. Cyril and Methodius

st. Rugjer Boshkovicj 16 Skopje, Macedonia

e-mail: {bojan.kostadinov, mile.jovanov, emil.stankov}@gmail.com

Abstract. Data collection and machine learning are changing the world. Whether it is medicine, sports or education, companies and institutions are investing a lot of time and money in systems that gather, process and analyse data. Likewise, to improve competitiveness, a lot of countries are making changes to their educational policy by supporting STEM disciplines. Therefore, it's important to put effort into using various data sources to help students succeed in STEM. In this paper, we present a platform that can analyse student's activity on various contest and e-learning systems, combine and process the data, and then present it in various ways that are easy to understand. This in turn enables teachers and organizers to recognize talented and hard-working students, identify issues, and/or motivate students to practice and work on areas where they're weaker.

Keywords: students, platform, STEM, data, e-learning, programming contests.

1. Introduction

Science, Technology, Engineering and Mathematics (STEM) are academic disciplines that have a huge potential to improve competitiveness and spur economic development. Nowadays, many countries are working very hard to change their current education policy, in order to respond to the increased demand for occupations related to science. Even more, the topics related to STEM are important for other jobs and aspects of life – as creative thinking, proofs and observations can be successfully utilized in various fields.

A number of researchers have been working on developing innovative ways to motivate students and help them succeed in STEM (e.g. Hall *et al.*, 2011; Hossain and Robinson, 2012; Wang and Degol, 2013; Joyce, 2014; Kearney, 2016). With the increase in popularity of various open-source software and code-repository hosting solutions, schools and teachers have a relatively easy way of adopting software in education. Likewise, there are open websites and systems that offer tutorials, documentation and tasks that anyone can freely use. In this paper, by mainly focusing on informatics and algo-

rithmic thinking, we'll discuss the usage of libraries and tools for gathering data and statistics from various e-learning platforms and websites, as well as a way to process that data to produce meaningful results. For example, by analysing student activity, we can identify issues such as lack of motivation or inadequate use of time – thus enabling us to help the affected students succeed better.

The paper is organized as follows. In Section 2 we will discuss education, STEM and competitions in informatics. In Section 3 we focus on various contest and e-learning systems for teaching informatics and preparing students for competitions. We will also describe several popular websites that organize algorithmic contests. Section 4 contains information regarding data collection, consent and statistics, while the following section builds up on the facts and examples presented thus far, and describes a project called Hero which is currently used actively to collect and process data from various systems and websites. In Section 6 we summarize our findings.

2. Education and Contests

Numerous countries and international organizations recognize education as a basic human right. Although teaching may occur in both formal and informal settings, primary and (sometimes) secondary education is compulsory in most countries. Additionally, online and electronic platforms enable students and professionals to learn at their own pace. Computer science, programming and software development are some of the most popular fields to study online, and newer video courses and tutorials usually encompass useful exercises that students can practice on.

Competitions are another major factor in education (Verhoeff, 1997). They can help to encourage students to perform better in school (for example, to earn scholarships), to form teams and study groups with individuals that have similar interests, or to win various awards and certificates. Some companies use competitions to hire the best professionals or university students, while schools or accredited organizations use competitions to increase interest in a certain field or subject. National competitions are used to select students that will later represent the country at global competitions (such as, for example, the International Olympiad in Informatics).

In the list below, we present some of the most popular competitions in mathematics and informatics. In the next section we will present several websites and grading systems which are used to award points to solutions of algorithmic problems. Some of these websites, like Codeforces, have blogs or pages that contain tutorials that students can use to learn programming languages, algorithms or data structures.

- **The International Olympiad in Informatics** (IOI, 1989) is a popular programming competition that takes place each year. Countries can send teams of 4 contestants, which compete by solving various algorithmic tasks. The IOI uses an automated grading system, and most of the tasks require students to write programs in one of the allowed programming languages.
- **Bebras** (Bebras, 2004) is a competition that is organized in several countries, with the main goal of promoting informatics, computer science and computational

thinking among pupils. Most of the tasks are multiple-choice, fill in the blank, or interactive problems. Compared to the IOI, which has a very low number of female participants, Bebras is actually popular among both male and female students. The number of participating students increases every year (see Fig. 1). The competition is typically organized by school teachers, during regular school hours. Countries use different (custom) grading systems to organize the event, usually developed by the organizer or a partner.

- **The International Math Olympiad** (IMO, 1959) is an international competition in mathematics that takes place in a different country every year. The first event took place far back in 1959 in Romania. Countries send teams of up to six students to represent them at the prestigious IMO event. Google sponsored \$1 million to the organization behind the International Math Olympiad in 2011.

In Macedonia, regional and national competitions are organized in several STEM-related subjects. For example, competitions in informatics are held in Macedonia since 1990, and they involve solving algorithmic tasks by coding solutions that are later automatically graded (Kostadinov *et al.*, 2015). After several stages of competitions, the best primary and secondary school students represent the country and themselves at the Balkan Olympiad in Informatics (BOI), the International Olympiad in Informatics (IOI), and the Junior BOI (for younger students).

Starting recently, there is also a separate regional and national competition for primary school students in computational thinking, using Bebras-like tasks. This has led to an increase in the participation at the national competitions in informatics, and an improvement in the gender balance (due to more female participants). Awards are given to the best students from each grade (age group). Students that win a medal at an international competition, as well as their teachers, are awarded a monetary prize by the Ministry of Education of Macedonia.

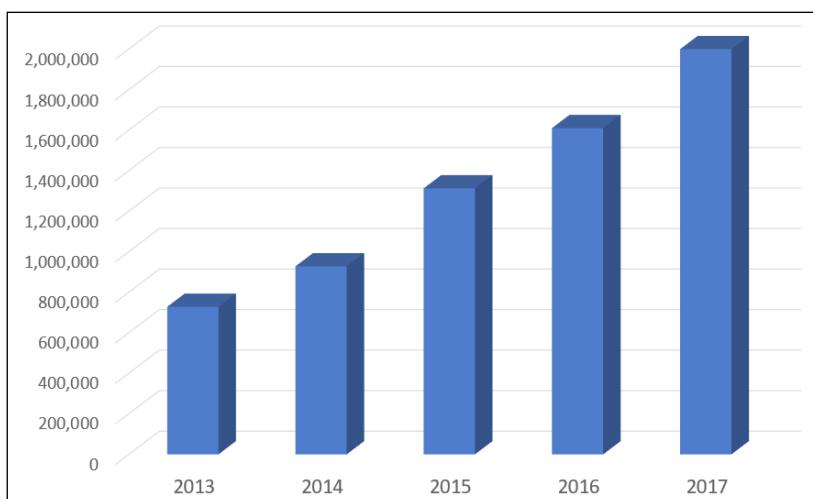


Fig. 1. Number of participants at Bebras.

3. Grading and E-learning Systems

E-learning has a lot of advantages compared to traditional learning. In most cases, when schools or teachers discuss distance or e-learning, they talk about blended learning – or a combination of the traditional classroom with electronic technology. However, with the recent rise of popularity of Massive Open Online Courses (MOOCs), students and professionals are more and more likely to take courses completely through the Internet. A number of online platforms offer both free and paid courses in various areas: examples include Coursera (Coursera, 2012) and Udacity (Udacity, 2011). Videos, animations, documents, forums and quizzes are just some of the features that are offered by online courses.

E-learning doesn't have to involve a traditional teacher with a degree in education. Some popular websites also offer courses created by professionals in different fields, so e-learning is used by students of various ages – from primary school students (where the school or teacher uses electronic technology in addition to traditional learning), to professionals (who are using online courses to acquire new skills). It's believed that the rise of e-learning, the Internet and technology will help expand access to education to both the general public and businesses. Another big advantage compared to traditional learning is that online courses allow people to study at their own pace.

Software development is one of the fields where e-learning can help the most. Various platforms exist where people can learn by reading tutorials, watching videos, executing code and instantly viewing results, solving smaller tasks or running database queries with automated feedback and more. Competitions in informatics also use grading systems to automate the process of scoring solutions – much quicker and with less chances of error (a mistake can still be made if the grading system experiences issues, or if the tasks and their scoring criteria aren't defined properly).

Table 1 shows examples of a few popular websites where software engineers go to learn, practice or compete. In this paper, we're interested in all these types of systems, as we want to track how students learn or practice STEM.

Since the main topic of this paper is the collection of data and statistics, it's important to understand what students and teachers do on these systems. Without the necessary

Table 1
Examples of systems used by programmers

Product	Type	Features
Udemy	Online learning platform	Video tutorials, discussions, quizzes, file sharing
Stack Overflow	QA site	Questions/answers, learning by viewing best answers on common questions
CMS	System for running programming contests	Grading solutions, answering questions, administering contests (self-hosted)
Codeforces	Site with contests & tutorials	Contests, practicing in virtual competitions, learning from tutorials and posts
MENDO	Site with contests & courses	Site with forum, wiki, national contests and courses for C++ and algorithms

information, it would be impossible to decide what data to gather, process or store. Additionally, even if we had the option of downloading everything, it would still be important to know how to merge that data with the other information from a user profile.

Codeforces, for example, allows users to participate in competitions organized on the platform. However, it also enables them to read tutorials on various algorithms and data structures, and practice by solving tasks given at previous competitions. When a user solves a task on his computer, he can send it to Codeforces, and the system will automatically grade it and show the verdict.

The platform contains hundreds of tasks (available in what's called the "Problem-set"), and all of them are marked with the appropriate tag referencing an algorithm, technique or a data structure (e.g., "binary search"). Students may practice by solving tasks at random, based on their difficulty, or based on a tag.

Topcoder, a similar site to Codeforces, also organizes competitions and provides a section where users can practice. Additionally, it has data science tutorials on various algorithms and data structures. Tasks are similarly marked with their type and difficulty. By looking at a user profile, it's possible to see his rating and the latest competitions he participated in (i.e. his activity). It's important to note that Topcoder hosts other types of competitions as well – design, development, bug races, and more.

On the other hand, the grading system used at the International Olympiad in Informatics, referred to as Contest Management System (**CMS**), is mostly used as a pure contest management system. Organizers will upload tasks to the system, and students will later participate in a contest by solving those tasks. Looking at a scoreboard, it is possible to see the rankings and the results of each contestant for each task.

The accredited organization in Macedonia for conducting competitions in informatics for primary and secondary school students on national level is the Computer Society of Macedonia. A competition management system called **MENDO** is used to teach students about programming and algorithms, and to prepare students for competitions in informatics (Kostadinov *et al.*, 2010). In the last couple of years, MENDO has been used to organize almost all national competitions in informatics, as well as several Balkan and Junior Balkan Olympiads in Informatics. In total, more than 300 events have been organized, and more than 10,000 users have submitted around 350,000 solutions.

A forum, wiki and a commenting system allow users to ask questions, share solutions or discuss programming-related topics. A number of university courses have been organized on the platform by utilizing the various features and grading options.

Tasks with automated grading are a very powerful way to test the user's knowledge and to provide an opportunity for practice. However, they sometimes influence students in a negative way – for example, a user being unable to send a solution because he is not following the system instructions (printing data in a wrong format, etc.). MENDO is an interactive system with various feedback features, one of which is the ability to discover issues like the one mentioned previously – i.e. it runs a solution-specific analysis and matches issues with a preprogramed set of mistakes. Other options like the ability to download test cases, to view solutions, and to read open tutorials (with animations and interactivity), are also very useful and allow administrators to attract new participants without much involvement from their schools or teachers.

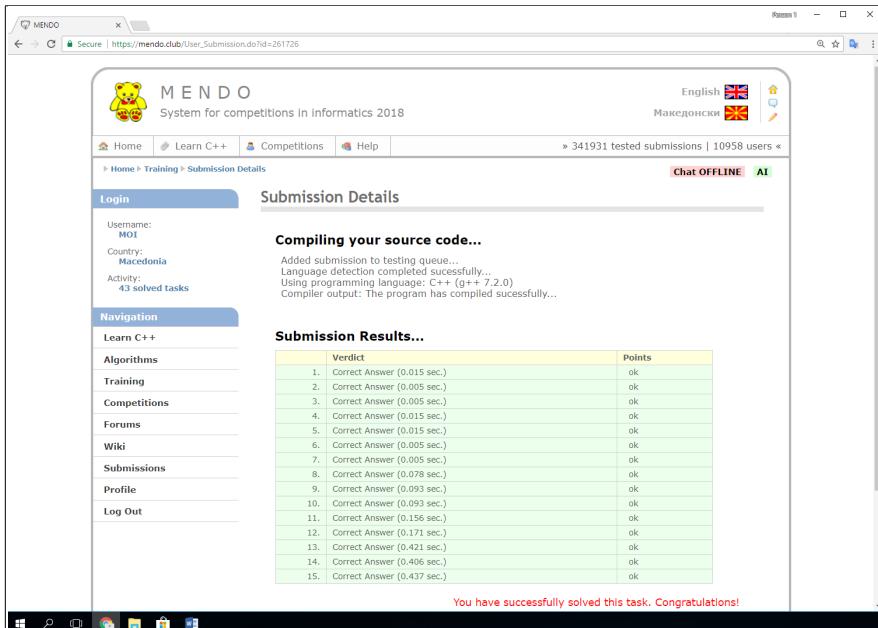


Fig. 2. Screen with submission details on MENDO.

On MENDO, students also participate in competitions and other events (e.g., university courses). During a competition, participants can (optionally) receive feedback – depending on how an administrator wants to configure each task. After a competition or lab exercise has completed, students (and other visitors) can view the results of the competition. This means that, if we want to gather this data for a system that tracks or combines user activity, we can do that easily since the information is readily available.

It's worth pointing out that there are many other systems and websites that software developers use. Some of them are used to organize events like Beaver – where students solve various interactive tasks. These are mostly open websites which allow students to view their results online. The Macedonian system (talent.mk) allows teachers and school principals to view all results, while students can only view their personal score (and download their certificate).

Some systems are developed and owned by companies, and are used to organize competitions such as Google Code Jam or Facebook Hacker Cup. Students and professionals participate in those competitions by registering on the event's website. For Code Jam, for example, results are then published on the official website, but there are other platforms which track that data and publish it in a more readable format (for example, filtered by country, language or round). It is possible to view the relative rank of contestants versus others from the same country, view statistics about the programming languages that contestants use to solve tasks, and more.

Other systems have been developed by universities to organize their internal courses, quizzes or admission tests. These are mostly available via the Internet (in order to allow students to submit homework, and for teachers to view results or add tasks), but sometimes they might be behind a university firewall.

4. Data and Statistics

Data science and analytics are used by various products and websites to gain valuable knowledge about users and their behaviour. With the power of computer science, statistics and artificial intelligence, software solutions can describe models and predict risk or performance. Analytics are used in all fields of study – social sciences, legal, business, medicine and more.

For web applications, it's common to use a premade solution like Google Analytics to both collect data, visualize it, and provide various other insights. Developers insert a small piece of code on a website, and the data instantly becomes available and visible on a dashboard. User and session counters, bounce rates, real-time reports, traffic sources, referrals, demographics and locations are just some of the information that can be easily tracked and analysed. Usually, other sources of data include server logs, the application's database (relational or non-relational), hosting providers' dashboards and APIs and more. Various free and commercial software solutions exist to parse this information, compare it side-by-side, and then visualize and group critical data.

It's important to realize that users need to be informed on how their behaviour and data is tracked and stored. Privacy laws in countries and regions must be respected, and most websites already have a Privacy policy that describes what data is stored.

But, users are spending time on other websites which might be out of our control. Some of these websites have privacy policies which allow the sharing of data, and have APIs that other software systems can use to look and download information they might be interested in. It might be beneficial to track, store, aggregate and present this data for users or organizers, and thus help them make better decisions. A good example of this is Codeforces, which has an API that can be used to monitor what tasks users are solving, contest standings, problem statistics, user's activity on contests and more. Each registered user on the platform has an API key which he can use to query information about him/her, but also info about other users and contest participants. Problem statistics can be used to discover easier and harder tasks, submission verdicts and resource consumption (like time and memory used).

The system used in Macedonia, MENDO, also has an API which is used to query and parse information. For each task, MENDO stores details about the number of users that attempted to solve the task, how many succeeded in doing so, details about the failed submissions (whether they were too slow or outputted wrong answers), and more. Additionally, for each user, the system knows which tasks were successfully solved, on which tasks the user failed, which competitions the user participated in, the lectures and tutorials that were completed and more. This data is mostly used by the students themselves when they connect to the website directly (outside of the API), as the system can tag (with a different color) each solved or failed task.

Most students who use MENDO like to see more information attached to each task or contest, as this helps them to make better decisions on how to spend their time on the system. Administrators constantly get requests for more such features. As an example, Fig. 3 presents some of the statistics and charts that are available after each

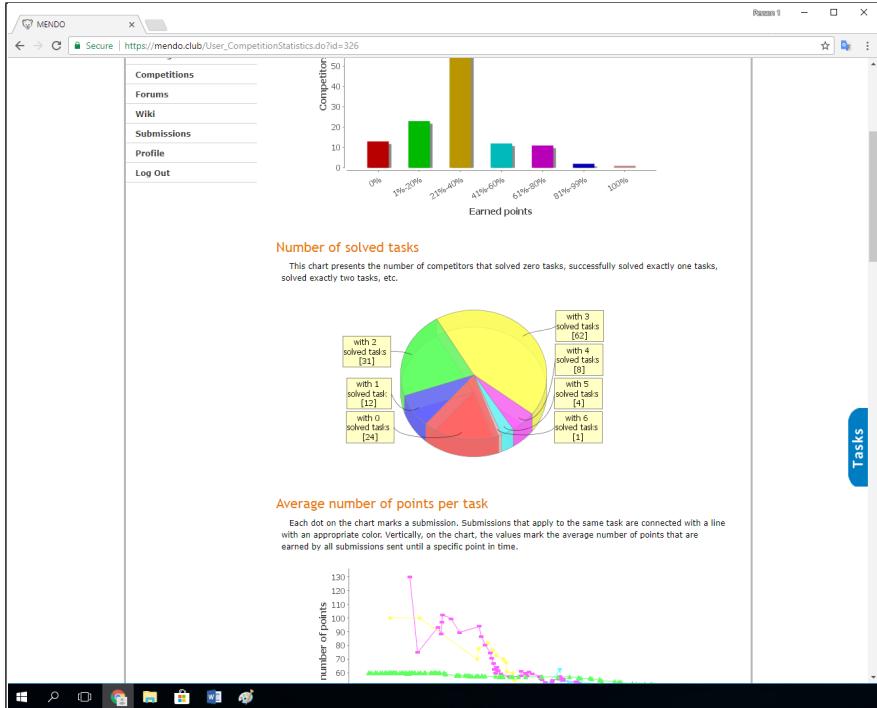


Fig. 3. Screen with contest statistics on MENDO.

contest. Others not visible on the screenshot include the points distribution, number of students that solved each task correctly or partially, the speed of the top performers and more.

Of course, this data is also used by contest organizers to see how well their tasks were designed, and whether the point distribution is too low or too high.

Before proceeding to the next section and presenting our software, we will outline the two most common ways of retrieving data from other websites, and the issues that need to be considered.

The first option is to use the official API of the website. There are a lot of benefits to this approach: structured data, documentation, adhering to the site policy on what data can be retrieved, and more. Most websites allow the data to be consumed in either JSON or XML format, and most programming languages have libraries already available. Since Application Programming Interfaces are meant to be consumed and parsed by machines, the system will already have rate limits in place, and will expect those endpoints to be accessed by bots.

The second option is to use web scraping – i.e. to extract data from the website by parsing the HTML code. Since websites are designed for humans, this data might not be structured and thus may be difficult to parse. Also, some websites employ methods to prevent web scraping. The biggest advantage of this technique is the ability to access everything that a regular user has access to.

With both techniques, it's very important to adhere to the site's policy on which data can be retrieved and stored locally. If no such information is available, it's common to ask the webmaster for permission and instructions. Although some webmasters might place restrictions on what load can be put on their server (so other users are not affected), most will allow scraping to occur if the platform is not a competitor product and you agree to link back to the original site.

After the data is transferred to a server, different database management systems can be used to store and analyse it. Depending on the size of the data, a relational or non-relational database can be used, or data can be stored with a cloud provider that offers storage services (and optionally, analytics).

Finally, it's worth pointing out the importance of respecting user's privacy – even when we store data which has been retrieved from other locations. Privacy is a major issue, and laws seem to be getting stricter by the year. If a person wants their data to be removed or hidden from a platform, that request should be fulfilled. With regards to programming and storing data about contest results and training activity, most software systems are open to users and they can then see their profile and use it to help improve (i.e. plan what to practice on next). Organizers can also use the data to track what students need to work on, and whether or not they are actively practicing and participating in competitions.

5. Software

In the previous sections, we outlined several systems that are used by contestants in algorithmic programming. Additionally, we presented a couple of ways in which the activity on those systems can be collected. The Hero app is a software application that tracks, stores and visualizes that data, and which is currently used by several organizers and contestants in Macedonia.

The initial idea for a software solution like this came 1) from the work that organizers needed to do in order to track contestants after the national cycle (in order to help them perform better at international competitions), 2) to make sure a student that did a bad result at one competition isn't eliminated if his results at many other online systems are good, and 3) from various requests from contestants to learn more about the practice plans of students that performed better than them.

In the Hero app, contestants can be registered by configuring the usernames they use on various systems, or by searching for them by location (for example, contestant names on Codeforces can be listed by country). Additionally, it's possible to view rankings by country or group. One example of grouping contestants is listing the activity of everyone who participated at the Macedonian Olympiad in Informatics (around 20 students). This in turn enables users of the application to quickly determine what contestants are doing.

Currently, the application tracks activity on the following systems:

- mendo.mk
- topcoder.com
- codeforces.com

As an example, when looking at a user named “John Doe”, it’s possible to see how he performed at every competition from the national cycle (since all of them are organized on `mendo.mk`), the tasks he solved by practicing on `mendo.mk`, TopCoder or Codeforces, or his rank/results during TopCoder or Codeforces rounds.

Of course, if one user has problems with a certain task, he can view which other friends from his group solved the same one, and ask for help to understand the solution overview or source code.

It’s important to realize that several other websites offer the option to see the activity of other users, and to organize or track a personal practice plan. However, Hero’s performance, grouping and visualization abilities separate it from other similar solutions. On Fig. 4 you can see a screenshot of the Hero app, showing student’s activity on the MENDO contest and training management system.

A rating value is calculated for each contestant, depending on his recent competition results. Exact numbers like the count of solved tasks are also readily available. The application has several additional views showing a calendar of future events, tutorials, compact views of profile data and more. These can be used by both organizers, teachers and contestants to plan practice rounds and to stay informed of various online events. New tutorials can be added by organizers and are grouped and tagged by their complexity. A calendar lists various online events on popular websites. This section of the

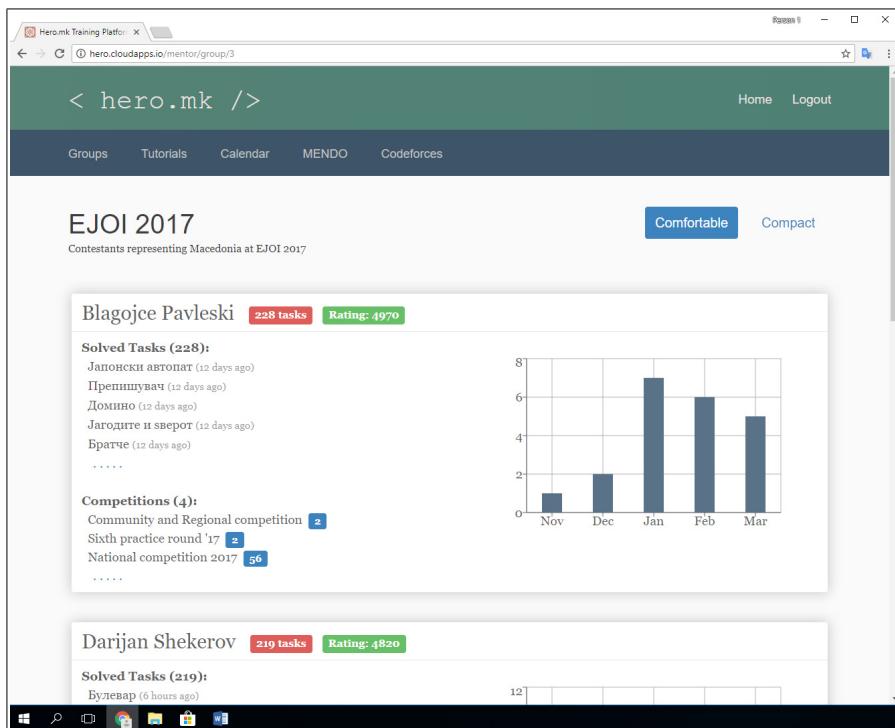


Fig. 4. Screenshot of the Hero app.

platform is populated automatically by downloading event calendars available online. The main idea is to enable students and teachers to subscribe to these events, and get notifications on the beginning of each week, so they don't miss an event they would otherwise be interested in participating in.

The implementation of the application uses both web scraping and APIs to get the necessary information. The fetching of data occurs on a schedule (every N minutes), in order to make sure the platform doesn't put unnecessary load on the systems it connects to. Additionally, there is a special database table that stores the time when every call occurred, to guarantee that a programming mistake won't lead to an unwanted denial of service attack.

From an implementation perspective, the Hero app runs on Node.js, a JavaScript runtime with a non-blocking I/O model that executes code server-side. Express.js is used as a web application framework, as it is fast and lightweight. This enables us to run the application on inexpensive virtual machines (as it's currently the case) or dedicated servers, with minimal resource consumption.

Hero stores all data in a PostgreSQL database, and Docker is used to run both the backend application and the database. PostgreSQL is an object-relational database management system which can run both on Windows and Linux. This means that all technologies that are used by Hero are open-source and free. One of the best features of PostgreSQL is the JSONB column type, which ensures that all stored data in the column is valid according to JSON rules, but enables applications to store data that might have minor differences (for example, there is some data available for Codeforces contests that isn't available for contests from MENDO). Hero attempts to store as much data as sent by outside APIs, even though some of that data might not be shown in the current version of the application.

Automatic backups to another server are scheduled every day, to guarantee that administrators won't have to execute scripts to poll the same data that was already downloaded. Most database management systems have the ability to create backups without locking or disabling the database.

The technologies used on the frontend include React and Recharts. React is an open-source library for building user interfaces, maintained by Facebook and Instagram. Large web applications use it to show data and forms without reloading pages. The main purpose of Recharts, on the other hand, is to help developers to easily create charts on React. Charts are the perfect way to present data that can be quantified, and Recharts enables the creation of line charts, bar charts, pies, radar charts, tree maps, scatter plots and more.

Hero can only be accessed by authorized users (students, teachers or organizers). New users can be added by administrators through the application. Because authorization is needed, all profiles and combined data is private and only visible from inside the application.

Finally, it's worth pointing out that every retrieval call (either via an API or scraping) is defined in a separate file. By creating an application using this method, it's easy to add additional functionality in the future, and download data from more systems or websites. Every call is tagged with a system name or website url, and internal checks

common to all retrieval calls guarantee that no outside system will ever experience an undesirably high number of requests – even if a (new) developer makes a huge mistake when programming additional calls.

Various middleware checks and database constraints are used by the application to restrict the storage of flawed data in the database, or the addition of duplicate data. For example, even if a user changes his username on an outside system, the application will try to match him or her by their e-mail address, full name, and past practice and contest experience.

6. Conclusion

Countries and various organizations are investing a lot of resources into STEM education, since those disciplines are needed to improve competitiveness and speed up development. Similarly, pupils and professionals interested in computer science are using various e-learning systems and websites to stay informed of new technologies, practice and improve their skills and get help when the need arises.

The rise of informatics has lead to the formation of several national and international competitions, like the International Olympiad in Informatics or ACM-ICPC. Students use various grading systems and websites to solve algorithmic tasks and prepare for those competitions. Most of these systems offer tutorials, videos, or automated grading, and are filled with useful data on how some students practice or what knowledge is needed to solve each task. This data can be collected, processed and analysed to help provide recommendations to specific students on how they can use their time more effectively when learning or practicing, and to inform teachers and organizers of any lack of activity. Privacy must be considered, as these systems help create user profiles and can easily visualize user activity.

Data collection, analytics and machine learning can be used in all fields – including medicine, social sciences, education and commerce. Multiple companies are investing in research associated with this area, and several open and commercial software solutions exist to facilitate the analysis of data and to provide recommendations.

Acknowledgement

The research presented in this paper is partly supported by the Faculty of Computer Science and Engineering, at Ss. Cyril and Methodius University in Skopje.

References

- Codeforces.* <http://codeforces.com>
- Computer Society of Macedonia.* <http://zim.mk>
- Coursera* (2012). <https://www.coursera.org>
- Hall, C., Dickerson, J., Batts, D., Kauffmann, P., Bosse, M. (2011). Are we missing opportunities to encourage interest in STEM fields? *Journal of Technology Education*, 23(1), 32–46.
<https://scholar.lib.vt.edu/ejournals/JTE/v23n1/hall.html>
- Hossain, M., Robinson, M.G. (2012). How to motivate U.S. students to pursue STEM (Science, Technology, Engineering and Mathematics) careers. *US-China Education Review A*, 2, 442–451.
- International Olympiad in Informatics (IOI)* (1989–2017). <http://www.ioinformatics.org>
- International Challenge on Informatics and Computational Thinking (Bebras)* (2004–2017).
<http://bebras.org/>
- International Mathematical Olympiad (IMO)* (1959–2017). <https://www.imo-official.org/>
- Joyce, A. (2014). Stimulating interest in STEM careers among students in Europe: Supporting career choice and giving a more realistic view of STEM at work. European Schoolnet, Brussels.
- Kearney, C. (2016). Efforts to Increase Students' Interest in Pursuing Mathematics, Science and Technology Studies and Careers. National Measures taken by 30 countries – 2015 Report, European Schoolnet, Brussels.
<http://www.dzs.cz/file/3669/kearney-2016-nationalmeasures-30-countries-2015-report-28002-29-pdf/>
- Kostadinov, B., Jovanov, M., Stankov, E., Mihova, M., Stojkoska Risteska B. (2015). Different approaches for making the initial selection of talented students in programming competitions. *Olympiads in Informatics*, 9, 113–125.
- Kostadinov, B., Jovanov, M., Stankov, E. (2010). A new design of a system for contest management and grading in informatics competitions. *ICT Innovations 2010*, 87–95.
- TopCoder.* <https://topcoder.com>
- Udacity* (2011). <https://www.udacity.com>
- Verhoeft, T. (1997). The role of competitions in education. In: *Future World: Educating for the 21st Century: a conference and exhibition at IOI 1997*.
<http://www.ioinformatics.org/locations/ioi97/ffutwrld/competit.pdf>
- Wang, M., Degol, J. (2013). Motivational pathways to STEM career choices: Using expectancy-value perspective to understand individual and gender differences in STEM fields. *Developmental Review*, 33(4), 304–340.



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, University “Ss. Cyril and Methodius”, in Skopje. He is one of the organizers of the national competitions in informatics in Macedonia, and the Beaver event.



M. Jovanov is an assistant professor at the Faculty of Computer Science and Engineering, University “Ss. Cyril and Methodius”, in Skopje. As the President of the Computer Society of Macedonia, he has actively participated in the organization and realization of the Macedonian national competitions and Olympiads in informatics since 2001. He has been a team leader for the Macedonian team at International Olympiads in Informatics since 2006, and member of the IOI International Committee since 2015. His research interests include algorithms, future web, and e-education.



E. Stankov is a teaching and research assistant at the Faculty of Computer Science and Engineering, University “Ss. Cyril and Methodius”, in Skopje. He is a member of the Executive Board of the Computer Society of Macedonia, and has actively participated in the organization and realization of the Macedonian national competitions and Olympiads in informatics since 2009. Currently he is a Ph.D. student at the Faculty of Computer Science and Engineering. His research includes analysis of program code correctness using different techniques, and its application to e-learning.

Creating the Original Bebras Tasks by High School Students

Hiroki MANABE¹, Seiichi TANI²,
Susumu KANEMUNE³, Yoshiki MANABE⁴

¹*Hakuyo High School*

²*Nihon University*

³*Osaka Electro-Communication University*

⁴*The University of Tokyo*

e-mail: manaty2005@mh.scn-net.ne.jp, tani.seiichi@nihon-u.ac.jp,
kanemune@gmail.com, yoshiki-125.kanken-1@mh.scn-net.ne.jp

Abstract. The Bebras Challenge is an International Challenge on Informatics and Computational Thinking (CT). The goal of the challenge is to make students interested in Computer Science (CS) and CT. The authors let students participate in Bebras in regular Informatics classes at a high school in Japan. Not only involving the challenge, but we also implemented a learning activity which students create original Bebras-like problems. The learning activity aims to make students recognize that materials for algorithmic thinking are around them. Most of the students worked well and produced idea full problems. They created many great works. And some of them were selected as Japanese representative questions for the International Bebras Task Workshop by the Japanese Committee for the IOI, which conducts the Bebras Challenge in Japan. Some of them were used in the actual Bebras Challenge. In this report, we show the students' original questions and discuss the educational effect of this learning activity.

Keywords: Bebras Challenge, informatics.

1. Introduction

The Bebras Challenge (ICICT, undated) is an international competition in computer science (CS) and computational thinking (CT). The goal of the challenge is to make students interested in CS and CT. The Bebras tasks are consists of Informatics comprehension, Algorithmic thinking, Using computer systems, Structures, patterns and arrangements, Puzzles, ICT and Society. It was initiated in Lithuania in 2004. Recently, the competition is being spread through more than 40 countries and expected to reach more than 40 countries this year. One of the reasons that the Bebras Challenge

has been widespread is that the competition sets are composed of good tasks which can motivate pupils to be more interested in informatics topics. The International Bebras Organizing Committee developed criteria for good Bebras tasks (Dagienė and Futschek, 2008).

Good tasks:

- are related to informatics, computer science or computer literacy
- allow learning experiences
- can be solved in 3 minutes
- have a difficulty level
- are adequate for the age of contestants
- are independent of any curriculum
- are independent of specific IT systems
- have easily understandable problem statements
- are presentable at a single screen page
- are solvable at a computer, without other hardware, additional software or paper and pencil
- are politically correct should be funny
- should have pictures
- should have interactive elements (simulations, solving activities, etc.)
- should give immediate feedback

Following these criteria, experts in each country create tasks and propose to Bebras Task Workshop held annually. The members of Bebras Task Workshop discuss and select the tasks for Bebras challenge carefully. After the Workshop, selected tasks are translated to each native language. And the task set for the competition is adjusted. Therefore, the Bebras Challenge is composed of high-quality tasks. In Japan, Bebras has four age groups:

1. Benjamin: grade 5 to 6.
2. Cadet: grade 7 to 8.
3. Junior: grade 9 to 10.
4. Senior: grade 11 to 12.

Bebras tasks are suitable CT teaching material (Izu *et al.*, 2017). Authors used Bebras tasks as educational material too and incorporated the Bebras Challenge into a high school informatics curriculum. Moreover, we also implemented a learning activity called ‘Creating Bebras Task’ units, and students create original Bebras-Like problems. Dagienė reported that creating Bebras Task gave teachers an opportunity to learn informatics concepts deeper (Dagienė *et al.*, 2016).

We considered that even high school students would be able to learn informatics through problem creation. Through the observation of the attitude of the students solving Bebras tasks, we hypothesized that they would work well, and they would learn computer science subjectively by themselves. And we expected that they create good tasks which would be suitable for actual Bebras Challenge. In this report, we show their progress and tasks and evaluate learning effect of ‘Creating Bebras Task’ units.

2. Background

2.1. Informatics Education in Japan

In Japan, all high school students learn the compulsory subject “Informatics”, which consists of two optional subjects named “Information Study for Participating Community (Society and Information)” and “Information Study by Scientific Approach (Information Science)” (Kanemune *et al.*, 2017).

In the subject ‘Information Science’, algorithmic thinking is treated. However, many teachers feel difficulty in guiding them.

2.2. Timeline of Preparation and Conduction of Bebras Challenge in Japan

The Japanese Committee for the IOI (JCIOI, undated) conducts the Bebras Challenge as below.

1. Creating tasks for submission to Bebras Task Workshop.
2. Selecting Japanese representative tasks for Bebras Task Workshop.
3. Preparation for proposal for Bebras Task Workshop.
4. Participation in Bebras Task Workshop.
5. Selection tasks for the Bebras Challenge in Japan and translating them into Japanese.
6. Conduction of the Bebras Challenge.

3. Research Method

‘Creating Bebras Task’ units were implemented from 2014 to 2016 in ‘Information Science’ classes, and about 320 grade 10 students joined every year. In Japan, school year starts at April and ends in March, and there is a two week “winter break” including the end and beginning of the year.

The Bebras Challenge is held in ‘Bebras week’ in November. Students tried Junior tasks of the previous Bebras Challenges which were published on the website before Bebras week, and participated in the actual Bebras Challenge.

By teachers interview after the challenge, most of the students commented: “I enjoyed the Bebras.” In the class after the Bebras Challenge, the teacher explained the connection between each problem and topic of the computer science, and implemented ‘Creating Bebras Task’ units as homework during the winter break to the students.

3.1. Expecting to ‘Creating Bebras Task’ Units

The units had an aim for students’ learning of computer science through the implementation from 2014 to 2016, and the other aim is added in 2016. In actual Bebras, there are many tasks using familiar stories and backgrounds such as nature or living around us. Therefore, we considered that students would look for the task theme around them and inspire creativity. We expected that they would learn computer science by themselves to create tasks. We also hypothesized that watching other students’ tasks enhance learning effect. If he/she watch his/her friends’ task, he/she would solve it. Through solving other tasks and watching other topics of computer science, students would increase their knowledge. Therefore, we planned in a lesson to take time to watch other tasks.

Though committee members of JCIOI, which consists of university faculty members and high school and junior high school teachers, make task candidates for submission to BTW, students may have different views from computer science experts and be expected to create excellent tasks. Therefore, we chose good tasks among students’ tasks and proposed them to the meeting of JCIOI in 2014 and 2015. Some tasks were selected to submit BTW and used in actual Bebras Challenge. Since we realized that it is encouraging the students that there is the chance of being used in actual contests around the world, we showed the possibility to students as another aim of the units in 2016.

3.2. Instructions to the Students

He handed out an a4 size worksheet and instructed to write below items to each student.

- 1) Title, Question, Choice for answer, Age category.
- 2) Pictures to explain the problem.
- 3) Answer and explanation of the problem.
- 4) Explanation of the connection to computer science.
- 5) Time for work, Comment.

The teacher also indicated the evaluation points of the task.

- 1) Quality of the task (i.e. whether pupils can enjoy or not).
- 2) Originality.
- 3) Understandability.

In order to make students learn, the teacher introduced the following items.

- The previous Bebras tasks.
- The booklet for beginners in informatics.

The teacher informed the students that the tasks would be used as ‘Aim for education’ and ‘Aim for suggested tasks.’

3.3. Class after Suggestions of the Tasks

After the winter break, the students submitted Bebras-Like tasks they created. In each class, the teacher scanned them into one PDF file (about 40 pages) and handed out to the students. Moreover, the teacher made a booklet including excellent 15 tasks he selected. He handed out the booklet to all the students in the classes and explained selected tasks.

3.4. Suggestion to the Selection Meeting of Japan IOI Committee

We suggested excellent tasks to the selection meeting of Japan IOI Committee and asked judgements whether each task can be used as Japanese suggested tasks or not. In the meeting, experts discussed which task was suitable for all the candidate tasks including experts created and students created. The selected tasks were translated into English and suggested to the Bebras Task Workshop.

4. Results

4.1. Implementation Status

As a sample, we indicate a student task (Fig.1). The problem requires participants to think about RGB color representation by KADOMATSU, one of the Japanese symbol



Fig. 1. Example of students' Bebras-Like task.

used in the new year. The problem is understandable, and it was fun to think the answer. Therefore, it has high quality can be used as a Bebras task directly. In the time entry, ‘4 hours’ was written as the time for work. And in the comment entry, ‘It spent much time to decide a type of the problem’ and ‘I thought I would like to doubt solutionists’ were written.

Many other students worked well and suggested high quality tasks. The mean time to work was about 1.8 hours, and the maximum was 12 hours. Topics as informatics were diverse. Cipher, Barcode, Binary Numbers, Programming, Math(Probability and Combination) were popular.

4.2. Aim for Education

To evaluate the educational effect of ‘Creating Bebras Task’ units, we used two data sets. One set was comments data they suggested. The other set was questionnaire data after watching tasks those other students created.

4.2.1. Creating Task Process

In the process of creating tasks, we show how students learned and wrote their impressions.

First, we asked following questions:

- 1) Did you learn computer science by yourself?

Yes 246 (77.4%)	No 72 (22.6%)
-----------------	---------------

- 2) How did you learn? (Multiple answers are possible)

1. Previous Bebras Tasks 179
2. Web Pages 95
3. Books 85
4. Others (ie. Asked family) 36

Next, we analyzed student comments written in prints. The most frequently used word was ‘difficult’. 146 students (46%) used it. The reasons were ‘To get an idea’, ‘To explain in understandable terms’ and so on. Second frequently used word was ‘interesting’. 87 students (27%) were used it.

Some comments are below:

1. It was difficult to think what is computer science.
2. The time I was looked for task theme was most interesting.
3. When I was gazing a clock, I suddenly got an idea. I realized inspiring.
4. After solving many Bebras Tasks, I began to create my task.
5. I felt creating task was interesting rather than solving. It was very difficult to write sentences understandable. I was interested in thinking how could I beguile others.
6. I worked very hard. After due consideration not to make some patterns of answers, I created my task.

4.2.2. Mutual Inspection of Submitted Tasks

In a lesson after the task submission, students watched many others' tasks. As a result, many students noticed that there were many topics in computer science field and realized that idea was important. 54 student (17%) commented on other students' tasks like 'What a wonderful Idea!' and 'Solving was fun'. Some comments are below:

1. I realized there were many topics in computer science around us. It was enjoyable to solve good tasks those the teacher selected. This lesson was a good chance to notice a connection between informatics and mathematics.
2. Creating task was challenging. But classmates created good tasks, and I understood that even high school student could create good ones by inspiration.
3. I was delighted!!!! My task was chosen as a good one.
4. Classmates' tasks were difficult than I expected. So, I was very interested in.

4.3. Good Tasks

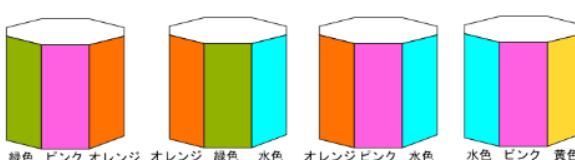
There are eight students' tasks those were submitted to Bebras Task Workshop as Japanese representative tasks, including ones selected as 'Favorite Task' in the workshop and were used in actual Bebras Challenge in several countries. In this section, we introduce four tasks used in Japanese Bebras Challenge and two tasks selected as 'Favorite Task' in the Workshop.

4.3.1. Four Tasks Used in Japanese Bebras Challenge

Task 1. Colorful Building

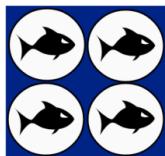
- ○ [Q] There is a colorful octagonal pillar (the left picture) building in the Bebras City. Each side wall of the building is painted with different colors, red, blue, yellow, green, purple, pink, orange and black.
- ×
- From the point A, you see the blue, purple and black walls.
 ● From the point B, you see the pink, yellow and purple walls.
 ● From the point C, you see the green, pink, and orange walls.

Choose the image which you can see from a point:



The Colorful Building task deals with the concept of permutation. To find out correct answer, organizing and classifying information are required. The student who was an author commented "I spent 30 minutes to create this. It was difficult to explain the problem in text."

Task2. Fish



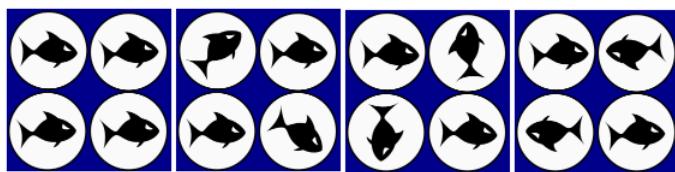
[Q] Four toy fish are placed as shown in the left picture.

If you turn a toy x degrees clockwise, the toy on the diagonal turns $(360 - x)$ degrees clockwise.

You operate as follows:

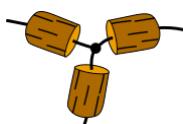
1. Turn the toy in the upper left 45 degrees clockwise.
2. Turn the toy in the lower left 90 degrees clockwise.
3. Turn the toy in the lower right 90 degrees clockwise.
4. Turn the toy in the upper left 45 degrees clockwise.

Choose the correct figure:



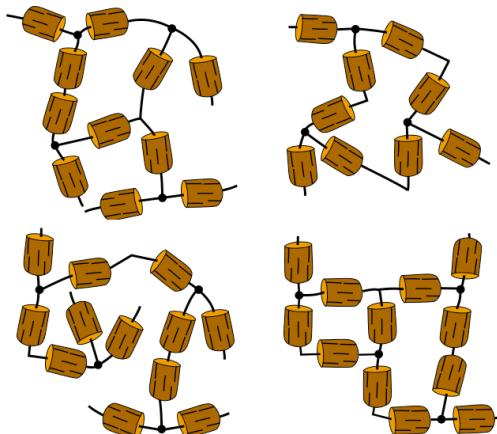
The Fish task deals with computer programs – sequences of instructions. To solve this problem using the arrows notation involves a lot of abstraction. The student who was an author commented “I spent 1 hour to create it. It was difficult to conceive the idea.”

Task3. Log Works



[Q] A beaver creates some works which consist of the log parts (the left picture). The log part consists of three logs and two ropes connected with each other. Therefore, the log part has three ropes which is connected to another log part.

What work the beaver cannot make from the parts?



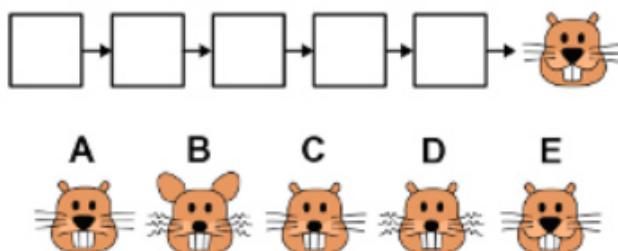
The Log Works task deals with the possibility of composition and combination using specific parts. Problems of construction such as ‘syntactic parsing’ are widely found out on informatics. A student who was an author commented “The idea came to me suddenly”.

4.3.2. Two Tasks Selected as ‘Favorite Task’ in the Workshop

At some Bebras Task Workshops, the members voted “Favorite Task” among all of the tasks. Many members selected the following two student tasks.

Task 4. Animation

[Q] B-taro is planning an animation, which shows a sequence of pictures of a face. The animation should run smoothly. Therefore, the order of the pictures is correct, if only one attribute of the face changes from one picture to the next. Unfortunately, the pictures got mixed up (picture below). Now B-taro must find the correct order again. Luckily, he knows which picture is last. He labels the five other pictures with letters A to E.



This task is introduced as sample task on the Bebras Web site. This task deals with data structure concepts, in particular with class which is a very important concept in object oriented programming (Dagienė, 2016).

Task 5. Sword and Shield

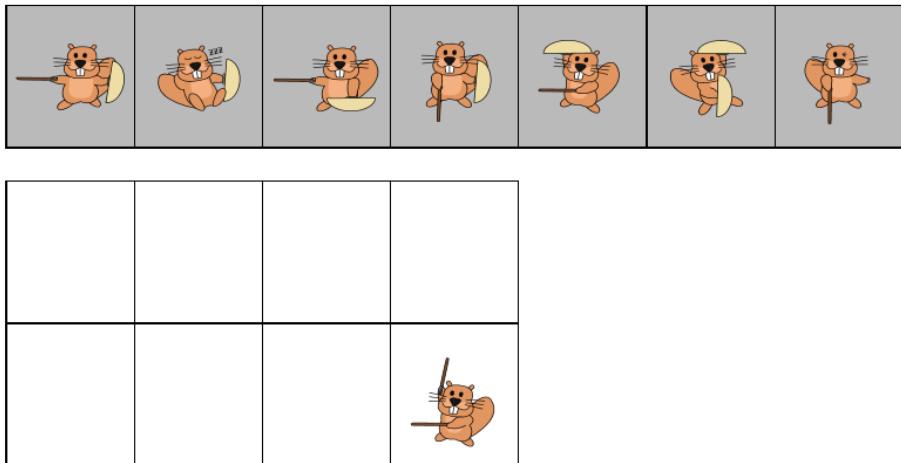
[Q] Luke Bevwalker adores swordsmanship. He attends a swordsmanship dojo. He finds 7 friends who also like swordsmanship. The eight beavers always train together. One day, the master of the dojo get them to strike poses as shown below:



The master says “you can make a legendary formation if you enter the boxes observing the following two conditions simultaneously:

- There is another beaver at the end of the sword.
- The shield is positioned to block the sword.

The master arranges the beaver (3) to the box (h).
 Which beaver is in the box (b)?



The Sword and Shield task is a complex puzzle. The backtrack algorithm which checks all the combinations requires an enormous amount of calculation. However, appropriate logical thinking could reduce it dramatically. A student who was an author commented “I tried various forms of conditions and branches to form a formation. If it has another solutions or not? I cannot verify it.”.

5. Consideration

We consider the effectiveness of ‘Creating Bebras Task’ units as following three points.

1. Students’ Efforts.
2. Influence to the learning of computer science.
3. Practical use to Bebras Challenge.

5.1. *Students’ Efforts*

Many students worked this activity seriously and created high-quality tasks. Some students created original tasks in short time by inspiration. Others created tasks in long time seriously. Through participating to the Bebras Challenge and trying to solve previous Bebras tasks, the students realized the features Bebras tasks have. They tried to direct the feature inside their tasks. Such considering were difficult but fun for them.

5.2. Influence on the Learning of Computer Science

By being imposed this learning activity in winter break, the students learned computer science voluntarily by using previous Bebras tasks or textbooks. Moreover, by watching other students' tasks, they could realize there are many topics in computer science field. The previous Bebras tasks and other students' tasks were made from items around them. Therefore, they would feel computer science close to them.

5.3. Practical Use to Bebras Challenge

Some students' tasks were submitted to the Bebras Task Workshop. They were used in Japanese Bebras Challenge and selected as "Favorite Task" at the Workshop. Therefore, we recognized that even high school students can create high quality tasks which could be used in actual Bebras Challenge.

'Creating Bebras Task' units were an activity to be able to contribute Bebras Challenge. For a student, the task he/she created was used in many countries and hundreds of thousands of pupils/students tried to solve it. We consider such dreamful learning activity was realized.

6. Conclusion

In a Japanese high school, we implemented a learning activity called 'Creating Bebras Task'. Creating task is more difficult than solving tasks. However, we gave an opportunity for the students to learn CS and made them feel the activity was fun. The future works of this research are to analyze the students' tasks more deeply and suggest as an educational method in CS.

References

- Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks, In: *ISSEP 2008: Informatics Education – Supporting Computational Thinking*. 19–30.
- Dagienė, V., Futschek, G., Stupurienė, G. (2016). Teachers' constructionist and deconstructionist learning by creating Bebras tasks: Constructionism in action, *Proceedings of Constructionism 2016*, 257–264.
- Dagienė, V., Stupurienė, G. (2016). Informatics concepts and computational thinking in K-12 education: A Lithuanian perspective. *Journal of Information Processing*, 24(4), 732–739.
- ICICT (undated). *International Challenge on Informatics and Computational Thinking*.
<http://bebras.org/>
- Izu, C., Mirolo, C., Settle, A., Mannila, L., Stupurienė, G. (2017). Exploring Bebras tasks content and performance: A multinational study. *Informatics in Education*, 16 (1), 39–59.
- JCIOI (undated). *Japanese Committee for the IOI*. <http://bebras.eplang.jp/>
- Kanemune, S., Shirai, S., Tani, S. (2017). Informatics and programming education at Primary and Secondary schools in Japan. *Olympiads in Informatics*, 11, 133–140.



H. Manabe is a teacher at Hakuyo High School. He received Ph.D. degrees from Osaka Electro-Communication University in 2013. He is a member of JCIOI since 2013. He has been working on programming education, algorithm education and statistics education.



S. Tani has been an executive director of the Japanese Committee for IOI (JCIOI) since 2009, and before this was a director of JCIOI since 2005. He received the BSc, MSc, and Ph.D. degrees from Waseda University, Tokyo, Japan, in 1987, 1990 and 1996, respectively. He is currently a professor at Department of Information Science, Nihon University. His research interests include computational complexity theory, computational topology, and complex network analysis.



S. Kanemune is a professor at Department of Electro-Mechanical Engineering, Osaka Electro-Communication University. He received Ph.D. in Systems Management from Tsukuba University. He is a director of JCIOI since 2016. He has been working on database, information system, programming education and robotics education.



Y. Manabe is an undergraduate student at The University of Tokyo. He had participated JCIOI annual events for kids. He developed Kanji (Chinese character) learning software and published a journal paper during he was in high school. His research focuses on Japanese Kanji education.

Combinatorial Property of Sets of Boxes in Multidimensional Euclidean Spaces and Theorems in Olympiad Tasks

Pavel S. PANKOV¹, Azret A. KENZHALIEV²

¹*International University of Kyrgyzstan*

²*American University of Central Asia*

e-mail: pps5050@mail.ru, azret.kenzhaliev@gmail.com

Abstract. Theorems (in general sense) are constituents of inventing, analysing and solving olympiad tasks. Also, some theorems can be proved with computer assistance only. The main idea is (human) reducing of primary (unbounded) set to a finite one. Non-trivial immanent properties of mathematical objects are of interest because they can be considered as alternative definitions of these objects revealing their additional features. A non-formal indication of such property is only initial data (size of domain) and only output data (proven/not proven) in a corresponding algorithm. One new and two known examples of such properties are considered, some techniques to convert theorem-proving algorithms into olympiad tasks are proposed.

Keywords: olympiads in informatics, immanent property, task, theorem, unboundedness.

1. Introduction

The aim of this paper is to propose computer-assisted search and proof of immanent properties of mathematical objects and to use such theorems in developing of olympiad tasks in informatics.

Theorems (in general sense, statements seemed to be true) are constituents of inventing, analysing and solving olympiad tasks. While authors of olympiad tasks are to describe all statements used for substantiation of their possible solutions, we suppose that contestants also think by means of some statements passing swiftly. We consider this item in Section 3.

Some well-known theorems were discovered by means of computational experiments or can be proved with computer assistance only. The main idea in such proofs is (human) reducing of a primary (unbounded) set to a finite one. We recount one new and one known examples of such theorems and hypotheses for Euclidean spaces of higher dimensions in Section 2.

Non-trivial immanent properties of mathematical objects are of interest because they can be considered as alternative definitions of these objects revealing their additional features. We hope that examples in Section 2 present such properties of Euclidean spaces. We propose the following non-formal indication of such property: an algorithm to prove it or any its corollary has none or only initial data (size of domain) and only output data (*proven/not proven*).

Some techniques to convert theorem-proving algorithms into olympiad tasks are proposed in Section 4.

2. Two Theorems on Immanent Properties of Euclidean Spaces with Unbounded Objects

We will consider Euclidean spaces R^N (N is a natural number) and “boxes” (parallelepipeds parallel to axes).

There are many results on “linear configurations” of finite sets of points on R^2 (see survey, Gardner, 1988, chapter 22), each of them can be considered as a theorem and an immanent property of a plane but they contain vast numerical conditions and are not “unique”.

Buddhist thangkas which do not “use” but “create” linear relations in planar finite sets of points can also be considered as revealing immanent properties of a plane but they are too complex.

We hope that the following problems are “natural” (Pankov, 2008) or have “short and elegant formulation” (Dagienė *et al.*, 2007).

We (Pankov *et al.*, 2005) put the problem on affine configurations without given quantities:

Problem 1. A finite set M is defined as follows (let its points be called M -points):

- 1) If two segments with endpoints being M -points have only mutual point then it is an M -point.

This condition is equivalent to the following (let convex hulls of non-empty subsets of M be called HM -sets).

- 1') If the intersection of two HM -sets is not empty then it is an HM -set.
- 2) The set M with any more point does not fulfill the condition 1 (1').

How many points can such set in R^N ($N \geq 2$) contain?

Consider the plane R^2 . As M is finite, there is a “basic” triangle which contains only three M -points (vertices). Exterior of such triangle consists of twelve plane sets: six rays and six infinite domains. Three of these domains cannot contain M -points obviously. Analysis of other nine sets is too complicated but the number of all possible cases is finite. We wrote an interactive program and proved that there exists only essential configuration and

Theorem 1. 1) The answer to Problem 1 in R^2 is only 6. 2) The configuration is the following: three points A, B, C and three points: B' on prolongation of the segment AB ; C' on prolongation of the segment BC ; A' on prolongation of the segment CA .

Also, we could construct a space model of an M-set of 8 points.

Hence, we put

Hypothesis 1. The space R^N has the immanent “finite-convex-hull”-number = $2N + 2$, $N \geq 2$.

The following statement would facilitate dynamical programming for sets of boxes.

Hypothesis 2. A set of $(N + 1)$ non-overlapping boxes in R^N can be separated by a coordinate hyper-plane (of dimension $(N - 1)$).

This is obvious for $N = 1$ and $N = 2$. Also, there are obvious examples stressing essentiality of this hypothesis:

Example 1 of four square boxes which cannot be separated;

Example 2 of three squares which cannot be separated by a straight line.

Hypothesis 2 seems to be too difficult to be proven for $N = 3$. Hence, we tried to involve computer.

To use computer successfully for proving theorems it is necessary to reduce a task to a finite search (see, for example, Pankov *et al.*, 2012).

Firstly, consider four equal cubic boxes in R^3 .

i) There are only two essential alternatives: projections of two cubes onto a coordinate plane are either overlapping or non-overlapping.

Hence, the task is reduced to consideration of integer cubic boxes with sides 2.

ii) Obviously, if any cube is far from others then a separating coordinate plane exists.

Specify this statement.

Lemma 1. If the convex hull of a projection of four integer cubic boxes with sides 2 onto a coordinate (for instance, “vertical”) axis is greater than 6 then a separating (“horizontal”) plane exists.

Proof. If this convex hull is greater than 6 then the gap between the projections of the “upper” and the “lower” cubes is greater than 2. If projections of two “intermediate” cubes do not fill the gap completely, then a separating plane exists; otherwise: if these projections overlap then a separating plane passes either over or under them otherwise between them.

Hence, it is sufficiently to consider arrangements of four cubes within a cube with side 6.

Such examination (of about 9 million arrangements, see Program 1

<https://cloud.mail.ru/public/MHLv/ktKFSxZ5H>) proved

Theorem 2. A set of 4 non-overlapping integer cubic boxes with side 2 within a cubic box with side 6 can be separated by a coordinate plane.

Applying Lemma 1 we obtain

Theorem 3. A set of 4 non-overlapping equal cubic boxes in R^3 can be separated by a coordinate plane.

This theorem corroborates Hypothesis 2.

By means of improving i) and ii) choppings-off there can be considered four non-equal cubes and general boxes in R^3 and five equal hyper-cubes in R^4 .

In other words, Hypothesis 2 may be reformulated as follows:

The space R^N has the immanent “separable-boxes”-number = $N + 1$, $N \geq 1$.

3. Theorems Related to Olympiad Tasks in Informatics

Such mathematical results represented as theorems can be classified as follows:

- Theorems invented or recollected to solve or to facilitate solving of the task (such as Lemma 1 above).
- Theorems proven by means of computer programs written for the task.

In their turn, theorems used by authors of tasks must be proven strictly to justify the author’s solution of the task. Mostly, theorems invented by participants during solving tasks pass swiftly, in implicit form without verbal formulation. It is enough to be assured in their validity for the participant (nevertheless, sometimes it is useful to write down any formulation to clarify the participant’s thoughts for themselves).

Remark. Sufficiency of the participant’s conviction on validity of an invented “theorem” depends on conditions of the competition. If results of testing programs are shown immediately to the participant (as it is in use at the ACM-ICPC International Collegiate Programming Contests and it was at National OI in Kyrgyzstan, March 2018) then the participant would submit the program based on this “theorem” without firm conviction; successive passing of all tests proves either validity of such “theorem” or its failing only in very exotic cases which were not covered in the set of tests.

If results of testing programs appear after the contest then the participant would be assured (in any way) in the validity of “theorem”.

As regards “theorems” to be proven by means of computer programs during a contest. Every correct program solving any correct task can be formally expressed as a “theorem” but with a too vast statement, including mathematical description of the set of initial data etc.

Some techniques to develop olympiad tasks on proving “intensional” theorems are proposed below.

4. Developing of Tasks of Type “to Prove a Theorem”

We will consider this item on examples of Theorem 1 and Theorem 2.

Firstly, one must not propose a task of type „*write a program to prove the statement ...*“ or „*write a program to check validity of the statement...*“ because the jury would have to check listings of programs submitted what is practically impossible.

Remark. A similar situation is at mathematical olympiads. A common type of tasks is „*to prove the statement ...*“ But contestants’ solutions of such tasks put a thankless duty for jury involving them into tangle debates and appeals: to prove that a submitted text is

not a complete proof (although it certainly contains parts of actual proof). We propose to convert such tasks into quantitative ones, as well as below.

Secondly, in our opinion, it is not convenient to propose tasks with responds of type „yes“/„no“ because there is probability of partially random guessing.

We propose to develop tasks with vast quantitative respond.

For example, Problem 1 may be put as

Task 1. Given a natural N in 2..10. How many sets M of integer points in the square $[-N..N] \times [-N..N]$ meet the following conditions (let their points be called M-points)?

- 1) The three points $(0,0)$, $(1,0)$ and $(0,1)$ are M-points.
- 2) If two segments with endpoints being M-points have only mutual point then it is an M-point.
- 3) The set M with any more integer point in $[-N-1 .. N+1] \times [-N-1 .. N+1]$ does not meet the condition 1.

Write a program which outputs this number $(\text{mod } 1000)$ (as usually, CPU time is 1 second).

Solving for $N=2$ and $N=3$ can be made by means of almost full search; solving for $N > 3$ demands improving of search, i.d. elements of proof (in mind) of Theorem 1. (For jury: answer follows immediately from Theorem 2: two options of three rays; only point on each of them).

The general idea of computer proof of a theorem of type $(*)$ “ $(\forall x \in X)(P(x))$ ” where X is an infinite set or a “too vast” one and $P(x)$ is a predicate is reducing $(*)$ to $(**)$ “ $(\forall x \in X_1)(P(x))$ ” where X_1 is a finite set accessible for a computer.

Hence, the following general task for using at contests on programming can be formulated:

How many $x \in X_1$ meet the condition $P(x)$? If the contestant would be able to write a corresponding program then the answer will be: all $|X_1|$. Then they may be congratulated: “You have proven the theorem “ $(\forall x \in X_1)(P(x))$ ” and ipso facto done the general theorem “ $(\forall x \in X)(P(x))$ ””.

For example, Theorem 2 (CPU time of Program 1 is about 36 seconds).

Task 2. Given an integer N in 4 .. 6. How many sets of 4 non-overlapping integer cubic boxes with side 2 within a cubic box with side 6 can be separated by a coordinate plane? (CPU time is 1 second).

To obtain full score the participant is to improve Program 1.

Some immanent properties can be also represented as “ $(\exists x \in X)(P(x))$ ” or “calculate $\min(\max)\{F(x) : x \in X\}$ ” with unexpected result.

For example, consider the Simpson’s paradox: there exist such positive integer numbers

$(***) A1 < B1, A2 < B2, A3 < B3, A4 < B4$ that

$(****) A1 / B1 > A2 / B2$ and $A3 / B3 > A4 / B4$ and $(A1 + A3) / (B1 + B3) < (A2 + A4) / (B2 + B4)$.

Task 3 (simple). Given N in 14..100. Find such $(***)$ that $(****)$ and $\max\{B1, B2, B3, B4\} = N$.

Task 4. Given N in 14 .. 100. Calculate the common fraction

$$\max \{ \min \{ A1 / B1 - A2 / B2, A3 / B3 - A4 / B4, (A2 + A4) / (B2 + B4) - (A1 + A3) / (B1 + B3) \} : \\ (***) , B1 \leq N, B2 \leq N, B3 \leq N, B4 \leq N \}.$$

5. Conclusion

We hope that computer-assisted search for immanent properties of mathematical objects would yield new intensional tasks being contributions to the mathematical science too and their solving would be interesting for participants of various contests on informatics and demonstrate them capacities of computers in scientific investigations.

6. Appendix – Task Spear

As gratitude to the hosts of the IOI'2018, we propose the following set of tasks for investigation.

It is known that Japan appeared as Drops into Ocean from Spear.

Let us try to optimize this process.

Task: given a binary matrix ('0's mean Ocean, '1's do Land) and the set of possible steps of Spear.

Initially Spear is over the NE corner of the matrix.

How many steps of Spear are necessary to create all Lands (to pass all '1's?)

The simplest sufficient set of possible steps is {S, W, E}.

Example: the matrix

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0
```

Possible beginnings of the optimal ways: WWSES... or SWWSE...

The answer is 32.

Until what size of the matrix can you construct an effective algorithm?

What other sets of possible steps ought to be considered (for example {S, SW, SE, W, E})?

What effective algorithms can be developed for such sets?

References

- Gardner, M. (1988). *Time Travel and Other Mathematical Bewilderments*. W.H.Freeman and Company, New York, 1988.
- Pankov, P.S., Alekseenko, S.N., Asanov, T.D. (2005). Interactive computer presentation of a configuration closed with respect to intersection of convex hulls (in Russian). *Bulletin of the Kyrgyz-Russian Slavic University*, 5(7), 85–88.
- 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. (2008) Naturalness in tasks for Olympiads in Informatics. (*Tasks and Training. Selected papers of the International Conference joint with the XX Olympiad in Informatics. – Cairo, Egypt*). *Olympiads in Informatics*, 2, 16–23.
- Pankov, P.S., Baryshnikov, K.A. (2012) Tasks of a Priori Unbounded Complexity. (*Selected papers of the International Conference joint with the XXIV Olympiad in Informatics. – Sirmione-Montichiari, Italy*). *Olympiads in Informatics*, 6, 110–114.



P.S. Pankov (1950), doctor of physical-math. 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, the leader of Kyrgyzstani teams at IOIs, 2002–2013. Graduated from the Kyrgyz State University in 1969, is a head of laboratory of Institute of mathematics of KR NAS, a professor of the International University of Kyrgyzstan.



A.A. Kenzhaliev (1999). Bronze medal at IOI'2016. Teacher Assistant at American University of Central Asia. Student of Korea Advanced Institute of Science and Technology (KAIST), Class of 2022.

How Hard Will this Task Be? Developments in Analyzing and Predicting Question Difficulty in the Bebras Challenge

Willem van der VEGT

Dutch Olympiad in Informatics
Windesheim University for Applied Sciences
PO Box 10090, 8000 GB Zwolle, The Netherlands
e-mail: w.van.der.vegt@windesheim.nl

Abstract. Predicting the difficulty level of a task on the concepts of computer science or computational thinking, like in the Bebras Challenge, proves to be really hard. Question difficulty breaks down in content difficulty, stimulus difficulty and task difficulty. Several instruments are suggested to predict the overall difficulty level, like using a questionnaire or a rubric; these instruments are applied on the data of a recent contest and proved useful. Relative scoring could also turnout helpful. Especially on content difficulty easy applicable solutions are lacking.

Keywords: Bebras contest, question difficulty, taxonomy, cognitive load theory.

1. Introduction

The Bebras Challenge is an annual International Contest on Informatics and Computational Thinking amongst the young (Bebras, 2018). Students from over fifty countries compete in their national contest. The questions used in these contests are chosen from a common task pool, which is composed in the Bebras Workshop where most of the contributing countries participate. The questions are formulated in a way that no prior knowledge is required.

The contest is about computer science and computational thinking; most of the tasks are categorized as ALP: Algorithms and Programming or DSR: Data, Data Structures, and Representations. A few tasks fit in the other three categories, CPH: Computer Processes and Hardware, COM: Communications and Networking or ISS: Interactions, Systems, and Society. Criteria for good Bebras tasks, using a former system for classification, are formulated by Dagienė and Futchek (2008). Dagienė and Sturupienė (2016) give an overview of current research on Bebras.

Contestants compete in their own age division. In the Netherlands contestants have 40 minutes to complete 15 tasks. These can be multiple choice questions, questions where an answer has to be given in the form of an integer or a short string, or interactive questions. The contest runs for a week in five different ages groups; some countries will also have an event for the youngest age group, 6–8 years, but the Dutch contest starts for grade 3; contestants are usually aged 8 year or (much) older. The best performing contestants for every age division are invited at a university for a second round (Beverwijkstra, 2018).

There are several reasons why it is important to predict the difficulty level of a Bebras task in advance (Van der Vegt, 2013). In a perfect world we would always be able to pretest questions to determine their difficulty and statistical characteristics before using them in a contest (Kibble and Johnson, 2011). But pretesting for a contest you really want to engage all possible students is hard to do. However, knowing the predefined difficulty level of a question is part of the contest. It is possible that contestants take this into account when answering a question.

Lee and Heyworth (2000) state that it is general agreed that students should be able to score higher in a test if the items or exercises are arranged according to their difficulty levels. They are looking for a measure of problem difficulty that can be obtained when a problem is created. They identify four different difficulty factors for algebra problems: the perceived number of difficult steps, the number of steps required to finish the problem, the number of operations in the problem expression and students' degree of familiarity with the question.

Leong (2006) explains the need to control question difficulty in test design in general. Test that contain too many easy or too many hard questions result in skewed mark distributions. And for comparison of tests through the years the distribution of item difficulty should be comparable. Lonati, Malchiodi, Monga and Morpurgo (2017) distinguish two main kinds of difficulties: on the one side intrinsic with the task, related to its content, and on the other side surface difficulties, depending on the task format and linguistic, structural and visual aspects. But Leong makes another distinction: he considers content difficulty, depending of the subject matter being assessed, stimulus difficulty, related to comprehending words and phrases in a test item and accompanying information, and task difficulty, referring to the work needed to formulate or discover the answer to the question. We will stick to his distinction.

In section 2 we will focus on content difficulty, including an enquiry on the possible role of taxonomies for this matter.

Section 3 handles with stimulus difficulty and possible reading problems.

Section 4 is dedicated to task difficulty, using cognitive load theory as theoretical background.

In section 5 we present a number of questionnaires, rubrics and procedures to think about question or item difficulty.

In section 6 we analyze a recent contest in the Netherlands, using some of the tools we found.

In section 7 we will discuss our findings and do some suggestions for future research.

2. Content Difficulty

Bebreas is about concept in computer science and computational thinking. Barendsen et al. (2015) show that it is possible to identify concepts on programming in various questions in the Bebras task pool. Izu, Mirolo, Settle, Mannilla, and Stupuriene (2017) describe how the goals of computational thinking are reflected in Bebras tasks. Lonati, Monga, Morpurgo, Malchiodi and Calcagni (2017) categorize a lot of Bebras tasks based on computational thinking skills.

The level of complexity of an assessment task is often determined using a taxonomy. The level of mastery is determined by the use of cognitive skills. Dunham, Yapa and Yu (2015) describe a way to use Bloom's taxonomy (Bloom, Engelbart, Furst, Hill and Krathwohl, 1956) for designing assessments in statistics education with varying difficulty levels. They focus on the depth of the thought process to solve problems, and they make explicit how to align assessment tasks on the taxonomy's scale.

Newman, Kundert, Lane and Bull (1988) concluded that students obtained higher scores for harder multiple choice questions when these problems were arranged in increasing cognitive order, i.e. knowledge, comprehension, application. For medium and easy question no such effect was found. But Kindle and Johnson (2011) observe that the assignment of learning taxonomies to multiple-choice questions has no relation at all to the difficulty of questions. They conclude that the categories in a taxonomy cannot be used to control exam difficulty.

Another issue that arises is the so called push-down effect (Merrill, 1971). A learner will attempt to perform a given response at the lowest possible level. For a novice a task can be highly demanding, while experienced students are able to push down the actual cognitive level on which they need to perform.

Finally we need to discuss the possibility to apply any taxonomy on a set of tasks that is aimed to test for insight in concepts, like in Bebras. In a school situation for many subjects reproduction can be a large part of an summative assessment. Bijsterbosch (2018) concluded for instance that in the Dutch lower level secondary education (the vmbo) over 60 % of all school exams questions on geography test reproductive skills. But Bebras provides a challenge where reproduction should be useless; the whole idea is to provide a set of tasks that do not require any pre-knowledge. This is a condition where the relation between learning objectives and the levels of mastery in a taxonomy is altered in a serious way. Adapting any form of taxonomy to this kind of contest will be needed before it will be possible to apply a taxonomy to the content difficulty of Bebras..

3. Stimulus Difficulty

Lumley, Routinksy, Mendelovits and Ramalingam (2012) created a scheme for describing the difficulty of reading items used in PISA. They compared the perceived and the empirical difficulty. They were able to conclude that their set of ten variables could be

reduced, because five variables explained about 57% of the variability in difficulty in items, and found indications that the variables in Table 1 contribute to item difficulty. Since reading and understanding a question is of course an important part of answering a task, these variables might prove useful, also for Bebras.

Remarkable is that their variable 7, Concreteness of information, which on its own correlated modestly but positively with item difficulty, was also found to be significant in the multiple regression analysis, but with a negative relation to item difficulty. Removing this variable lowered the explanatory power of the data. The authors suggest that if an item becomes more difficult, the degree of abstractness of the information readers need relates negatively to the items difficulty.

Lonati, Malchiodi, Monga and Morpurgo (2017) changed the formulation or presentation of some questions in the 2016-Bebras contest and presented these tasks to a new group of contestants. They report remarkable changes in the results for the altered tasks. On the task Recipe, which is on linked lists, the success rate was very low; in interviewing contestants they discovered that the text was not understood and generally read with no care. So they structured the problem in another way, added two ingredients and pre-filled three out of seven fields in the answer, while creating a new figure. They obtained a higher success rate in their control group and a significant decrease in discrimination.

Leong (2006) describes demands to increase stimulus difficulty. Use relevant technical terms, without elaboration or clarification, in the item; present information in such a way that requires candidates to do some re-organization. Supports that will decrease stimulus difficulty are for instance: Highlight or emphasize terms that require careful comprehension; tailor the resources to the task that candidates have to do.

Table 1
Revised PISA reading item difficulty scheme. Five most explaining variables
(Lumley, Routinksy, Mendelovits and Ramalingam, 2012)

3	Competing information	This refers to information in the stimulus and/or in the distractors (if multiple choice) that the reader may mistakenly select, or that the reader may generate, because of its similarity in one or more respects to the target information
5	Relationship between task and required information	The relationship between the question (the whole task, including the multiple-choice options where relevant) and the required information – that is, the kind of answer required to gain credit
7	Concreteness of information	The kind of information that readers must identify to complete a question
8	Familiarity of information needed to answer the question	This variable distinguishes tasks that focus on information inside or outside the text, or the text structure, that is close to the experience and concerns of the reader, from those focusing on what is likely to be remote and unfamiliar
10	Extent to which information from outside the text is required to answer the question	This variable deals with the extent to which the reader needs to draw on world knowledge, experience or personal beliefs and ideas and opinions in order to answer the question

4. Task Difficulty

One of the main concerns in question difficulty is the role of the working memory of a contestant. The working memory load is affected by the inherent nature of the material, the intrinsic cognitive load and the manner in which the material is presented. According to the cognitive load theory the limitations of the working memory are rarely taken into account in conventional instruction and assessment (Kirschner, 2002).

Conventional instructions tend to impose an extraneous cognitive load on the working memory, whereas learning something requires shifting from extraneous to germane cognitive load. Germane cognitive load is the effort that contributes to the construction of schemas. Schemata categorize information elements according to how they will be used. They can also reduce working memory load, since a schema can be treated as a single element. So schema construction aids the storage and organization of information in long-term memory and reduces working load memory.

In computer science education this process of schema formation has at least two effects: by building ever more complex schema by assimilating portions of lower-level schemas skills are developed, and once a particular skill is acquired, automatic processing can bypass working memory (Shaffer, Doube & Touvinen, 2003). Indications of working memory failures include: incomplete recall, failing to follow instructions, place-keeping errors and task abandonment (Shilbi & West, 2018).

Most of the available research on cognitive load is focused on education and instruction. Elliot, Kurz, Beddow and Frey (2009) apply cognitive load theory to test design. They present guidelines for testing; some for Bebras relevant suggestions are placed in Table 2.

Leong (2006) summarizes ways to decrease the task difficulty: decrease the number of steps in executing task; break up the task into a few sub-questions, order the steps such that they provide the scaffolding for subsequent steps. Increasing the task difficulty can be done by raising the number of steps in executing task, or present a task in which candidates need to devise steps to execute the task without cues and leaders.

Table 2
Recommendations for handling cognitive load in test design

-
- 5. Use bold for vocabulary words. Use red circles, arrows and highlighting for important elements of visuals
 - 6. Integrate explanatory text close to related visuals on pages and screens
 - 9. Text economy; all included visuals are necessary
 - 10. Don't add words to self-explanatory visuals
 - 13. Train test-takers in the test-delivery system prior to the test date
-

5. Instruments

Several tools have been developed to help test designers to predict the difficulty level of a question. In this section we will discuss two questionnaires, a rubric and a procedure. Since these are compositions with many parts, we will also try to analyze the ratio between content difficulty, stimulus difficulty and task difficulty.

5.1. *Questionnaires*

Earlier (Van der Vegt, 2013) we proposed a questionnaire for understanding and predicting the difficulty of a specific task. We focused on the question answering process, distinguishing reading, understanding, searching a mental representation, interpreting and composing, and the size of the problem. On the latter it is possible to give numbers, but the issues on the question answering process give more qualitative data. And weighting these data is a hard task in itself.

The questions I.a and I.b (Table 3) are about stimulus difficulty, I.c, I.d and I.e mainly about content difficulty and all questions II are handling task difficulty. This gives a 30/20/50 distribution on different kinds of question difficulty.

Vora, Jain, Mehta and Sankhe (2016) developed a method to assign weightage to question difficulty. Their instrument is shown in Table 4. The level of IQ is a subjective measure, the more the question makes sense, and the more it is related to the test subject, the more weightage is given. This is a measure for content difficulty. Length of question and pattern are merely on stimulus difficulty and question type is on task difficulty. So their questionnaire has a 25/50/25 distribution on different kinds of question difficulty.

The difficulty fraction is by definition the sum of the weights, minus the minimum total weight, divided by the difference of maximum and minimum weight, and if this

Table 3
Questionnaire for difficulty level estimation (Q1)

I.	The question answering process
a.	Which problems will there be in reading the question?
b.	Which problems will there be in understanding the question?
c.	Which problems can arise in searching the mental representation of the text?
d.	Which problems can arise when interpreting the answer?
e.	Which problems can arise when composing the answer?
II.	The size of the problem
a.	What is the number of elements in the question?
b.	What is the number of transformations for an element in the question?
c.	What is the number of constraints in the question?
d.	How do you rate the solution density of the problem?
e.	Will it be possible to solve the problem, using only your working memory?

Table 4
Weightage assignment (Vora, Jain, Mehta and Sankhe, 2016) (Q2)

Parameter	Weight range
Level of IQ (sense)	2–10
Length of question	2–10
Pattern	
a. Repetition of keyword	2–8
b. Image	0–2
Type of question	
a. True/false type	2
b. Simple MCQ	4
c. Calculated MCQ	6
d. Check Box (Multiple correct answers)	8
e. Text Box	10

fraction is below 0.1667 the question is defined as easy. A question with a difficulty fraction above 0.5 is considered hard. A medium question is neither easy nor hard. So a task that scores 31 as a total weight will have a difficulty fraction of $(31 - 8) / (40 - 8) = 0.72$ and such a task would be considered as a hard question. A task with a score of 8–13 points will be easy, any task with between 14 and 24 will be medium.

5.2. Rubric

At the Bebras Workshop 2018 the Italian team presented their work on a rubric, designed to make decisions on expected difficulty level (Bellettini, Lonati, Malchiodi, Monga and Morpurgo, 2018). For ten different aspects of the task they define three possible difficulty levels, and they distinguish between easy, medium and high difficulty. Their ten aspects are stated in Table 5.

For cognitive effort they specify an easy task as one that requires to understand a concept or a procedure and to perform a straightforward activity. A medium task is one

Table 5
Rubric lines (Bellettini, Lonati, Malchiodi, Monga and Morpurgo, 2018) (R)

-
- 1 Text and sentence length
 - 2 Familiarity of terms, notations, objects, and concepts needed to understand the task
 - 3 Consistency of terms and notations
 - 4 Other elements beside the text (pictures, diagrams, examples, etc.)
 - 5 Constraints, combinations, steps needed
 - 6 Relationships among objects to take into account
 - 7 Cognitive effort
 - 8 Use of notes or other supported material
 - 9 Solution space
 - 10 Solution check
-

that requires to analyze a context, a procedure or some properties. And a hard task requires to evaluate a setting, a system, a procedure, possibly comparing different hypotheses or strategies. Or it requires a creative effort to invent or build something.

In this rubric items 1, 2, 3 and 4 are about stimulus difficulty, 5 and 6 on content difficulty, and the other four mainly on task difficulty. So this rubric has a 20/40/40 distribution on question difficulty.

5.3. Procedure

Current practice is that the author of a Bebras task suggests the appropriate age category and difficulty level. At the international workshop, in selecting for a national contest and in translating the task adjustments may be made. In some countries, like in the Netherlands, the same task will be used for several age groups, assuming that the task prove to be more easy if the contestants are older.

Holmes and Read (2018) describe that it is very hard to make absolute judgements on question difficulty. They use a technique where a number of experts independent review many pairs of items and decide each time which item is more difficult to answer. This comparative judgement can be used to capture a group consensus well, and to avoid individual biases. This approach can be useful also for Bebras tasks; we took a set of six different tasks and asked a group of colleagues to order them from easy to hard. We scored the individual results from 1 (easy) to hard (6) and added the individual scores for each task. The total scores were a perfect match with the relative difficulty level. Kindle and Johnson (2011) report a similar practice: Each of nine faculty members was misjudging the difficulty level of some of the tasks in an exam, but the average score proved to be much better.

Bramley and Wilson (2016) asked a group of experts to estimate the mean marks for every question in a specific test, by using information on the results of previous test and looking for nearly identical questions. Statistical information can be used by the experts to guide their judgements. Developing a benchmark of used questions with known difficulty levels could be helpful in predicting the difficulty of new developed tasks.

6. A Recent Contest Analyzed

In an analysis after the contest we can define the actual difficulty level of each task as the p-value: item difficulty is simply the fraction of contestants taking the test who answered the item correctly. The larger the fraction getting an item right, the easier the question.

For the highest age groups of the first round of Bebras in the Netherlands in 2017 graphs of the percentages of good answers are presented in Fig. 1, Fig. 2 and Fig. 3. And though some sections of the questions are really well predicted, like the hard questions for age group IV and a few of the easy questions in all of these contests, a lot of questions

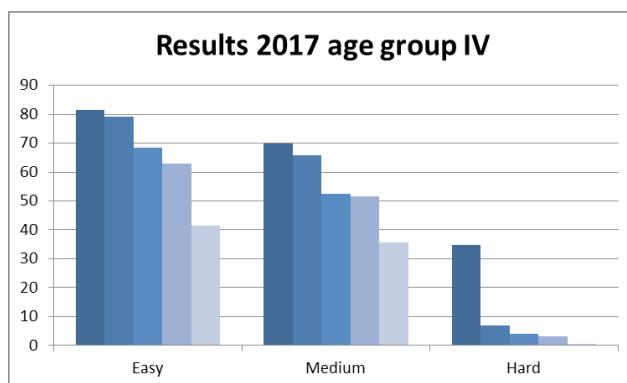


Fig. 1. Results of the 2017 contest for age group IV.

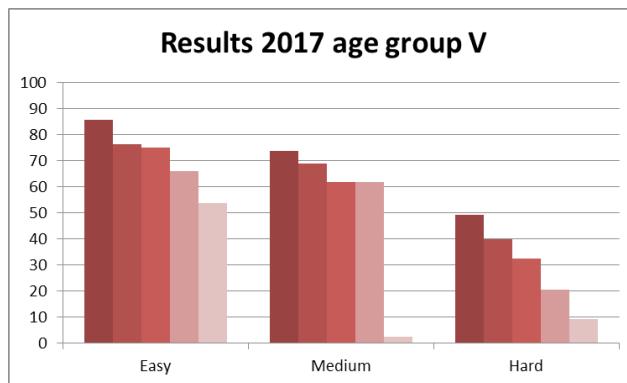


Fig. 2. Results of the 2017 contest for age group V.

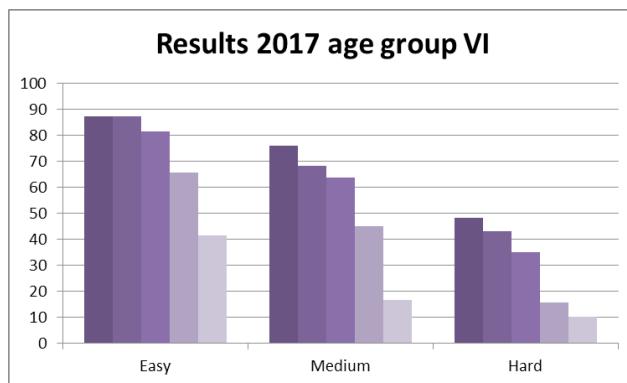


Fig. 3. Results of the 2017 contest for age group VI.

were under- or overestimated. An easy measure for the quality of our predictions is the percentage of misplaced tasks. In the contest presented in Fig. 1 this was 27%, in Fig. 2 it was 40% and in Fig. 3 it was 47%. Table 6 gives an overview of this measure.

The mean scores for the questions in a difficulty group are presented in Table 7. In all three contests the results are as expected, though the hard questions for age group IV turned out to be extremely difficult. The hardest question for age group V turned out to be a task we qualified as medium, and in age group VI the most difficult tasks of the

Table 6
Percentage of misplaced tasks in the first round of 2017

Year	Age division	Harder than predicted	Easier than predicted	Percentage misplaced
2017	IV (12–14)	2	2	27%
	V (14–16)	3	3	40%
	VI (16–18)	3	4	47%

Table 7
Results of the difficulty groups in the first round of 2017

Year	Age division	Easy questions	Medium questions	Hard questions
2017	IV (12–14)	66.72	55.13	9.93
	V (14–16)	71.39	53.86	30.29
	VI (16–18)	72.69	53.92	30.48

Table 8
Task analysis of age group VI in Dutch Bebras 2017

Task-ID	Assigned difficulty level	Success	Q1	Q2	R
2017-CA-12	Easy	87.42	0.40	0.22	0.30
2017-IS-01	Easy	86.37	0.40	0.28	0.35
2017-BE-05	Easy	81.62	0.50	0.31	0.40
2017-RU-03	Easy	65.70	0.55	0.38	0.55
2017-IR-07	Easy	41.39	0.70	0.47	0.60
2017-CA-07	Medium	75.88	0.60	0.53	0.55
2017-PL-02	Medium	68.17	0.65	0.59	0.60
2017-CH-01b	Medium	63.73	0.75	0.59	0.60
2017-CZ-04c	Medium	45.22	0.70	0.66	0.70
2017-CH-07b	Medium	16.59	0.85	0.63	0.80
2017-KR-07	Hard	48.37	0.75	0.66	0.70
2017-SK-12a	Hard	43.06	0.85	0.66	0.70
2017-UK-04	Hard	35.16	0.90	0.81	0.80
2017-KR-03	Hard	15.67	0.85	0.78	0.75
2017-SI-04	Hard	10.12	0.90	0.63	0.70

section easy and the section medium were harder than predicted. In section 6.1 we will analyze some of these tasks for age group VI more in detail.

Izu, Mirolo, Settle, Mannilla, and Stupurienė (2017) make a similar analysis for the 2014 and 2015 contest in five countries. They use the rank match to see how well the difficulty level was predicted; the outcomes are between 40% and 72%. This corresponds to 100 minus the percentage misplaced, so our Dutch values for rank match are 53%, 60% and 73%.

We did an analysis for all 15 questions of the highest age group in the Dutch Bebras 2017, using the three instruments in section 5. Q1 refers to the questionnaire of Table 3; all questions are scored as 0 for easy, 1 for medium or 2 for hard, so the total is on a scale from 0 to 20. Reported in the table is the fraction of the maximum score. Q2 stands for the questionnaire of Table 4; the result reported is the difficulty fraction. And R is the Italian rubric, presented in Table 5, also scored using 0, 1 or 2 per item and presented as a fraction.

At first sight all three instruments can be used to order the tasks from easy to hard. For each of the three scales we performed a linear regression, the results of which are seen in Fig. 4. We also calculated the correlation coefficient. In all three cases this coef-

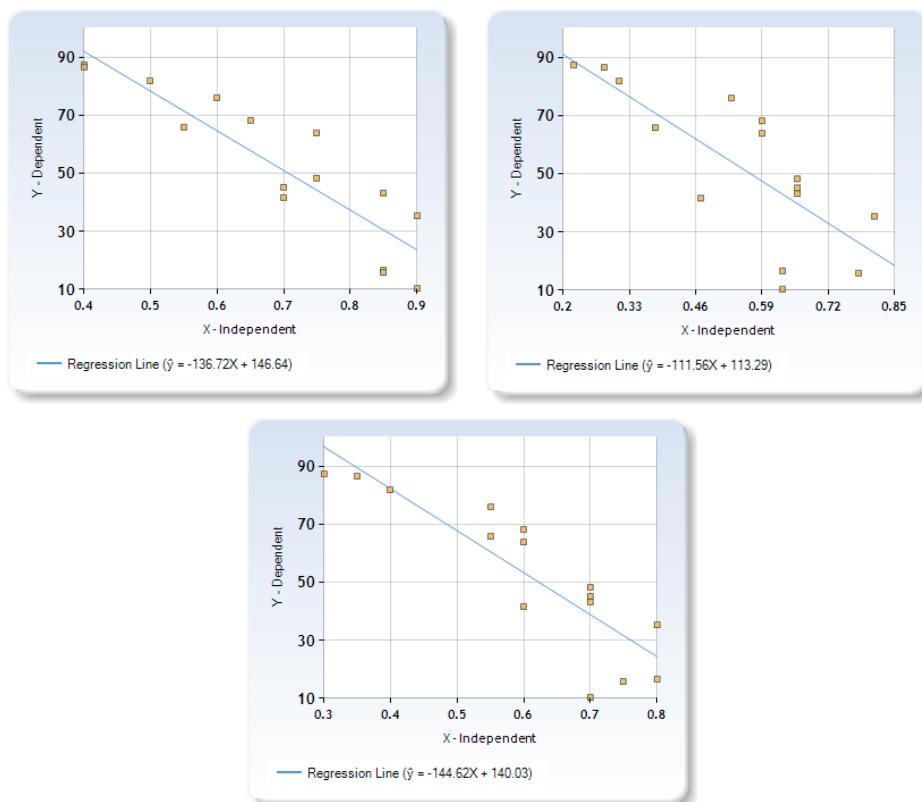


Fig.4. Lineair regression of Q1 (left), Q2 (right) and R (bottom) and success-rate for Bebras 2017.

ficient was almost -1. Q1 had a correlation coefficient of -0.90, Q2 one of -0.77 and the result for R was -0.87. So the rule of thumb can be that the higher the difficulty fraction, calculated with either of these instruments, will be, the harder the question will prove in practice.

Questionnaire 2 proved the hardest to use for a scorer. Level of IQ (sense) asks for a number on content difficulty; we used the assigned difficulty level, with a 3 for an easy task, a 6 for a medium one and a 9 for a hard task. But these judgements were of course already about more than content difficulty. So this measure has a systematic flaw. For questionnaire 1 and the rubric a lot of close calls had to be answered; the rubric has specifications when to assign a specific score for a task, but these specifications are not always decisive enough. Questionnaire 1 is lacking these specifications at all, so scoring is quite intuitive, but could easily be biased, since the actual results of the contest were already available.

One of the main factors in the actual use of this kind of instruments will be the time a scorer needs to answer all questions. The items should be well-defined and the boundaries between possible scores need to be clear; otherwise these instruments will never be used in designing an actual contest because no one has the time to fill in the forms.

7. Discussion

Application of several tools, developed to predict the difficulty level of a (Bebras) task, can help the contest designer to create a fair, balanced contest. All three investigated instruments can be used for this goal. In further research one could look to the best balance for the components and weights on content, stimulus and task difficulty.

Content difficulty is the most unclear item in predicting difficulty. More research is needed on the use of taxonomies, especially for questions that do not use any pre-knowledge, or other systematic approaches to identify content difficulty.

The use of procedures for relative scoring seems promising. Combining individual judgements on question difficulty can improve the overall decision. Integrating the use of questionnaires and relative scoring based on the output by several scorers will be a valuable condition for a more systematic preparation of this part of the contest.

Testing and the need to predict task or question difficulty go beyond the boundaries of Bebras. A lot of recent research on cognitive psychology shows that stimulus and task difficulty play an important role in the performance of contestants. Instruments used to predict question difficulty should include these insights.

References

- Barendsen, E., Manilla, L., Demo, B., Grgurina, N., Izu, C., Mirolo, C., Sentence, S., Settle, A., Stupurienė, G. (2015). Concepts in K–9 computer science education. In: Dagienė, V. (Ed.), *ITICSE '15 : Proceedings of*

- the 2015 ACM Conference on Innovation and Technology in Computer Science Education Conference, 2015 Vilnius, Lithuania – July 04–08.* 85–116.
- Bebras website (2018). <http://bebras.org/>
- Belletini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A. (2018). A rubric to help with Bebras tasks. Presented at the Bebras Workshop 2018, Protaras, Cyprus.
- Beverwedstrijd (2018). (in Dutch) <http://www.beleverwedstrijd.nl/>
- Bijsterbosch, H.D. (2018). Professional Development of Geography Teachers with Regard to Summative Assessment Practices. University of Utrecht.
- Bloom, B.S., Engelbart, M.D., Furst, E.J., Hill, W.H., Krathwohl, D.R. (1956). Taxonomy of Educational Objects: The Classification of Educational Goals, Handbook I: Cognitive Domain. New YorkL David McKay Co Inc.
- Bramley, T., Wilson, F. (2016). Maintaining test standards by expert judgement of item difficulty. In: Research Matters, Issue 21.
- Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: R.T. Mittermeier and M.M. Syslo (Eds.), *ISSEP 2008, LNCS 5090*. Springer-Verlag Berlin Heidelberg, 19–30.
- Dagienė, V., Sturupienė, G. (2016). Bebras – a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 15(1), 25–44.
- Dunham, B., Yapa, G., Yu, E. (2015). Calibrating the difficulty of an Assessment Tool: The Blooming of a Statistics Examination. *Journal of Statistics Education*, 23(3).
- Elliot, S.N., Kurz, A., Beddow, P., Frey, J. (2009). *Cognitive Load Theory: Instruction-based Research with Applications for Designing Tests*. Presented at the national Association of School Psychologists 39; annual convention, Boston, MA.
- Holmes, S., Rhead, S. (2017). A level and AS mathematics: an evaluation of the expected item difficulty. Ofqual/18/6344.
- Izu, C., Mirolo, C., Settle, A., Mannilla, L., Stupurienė, G. (2017). Exploring Bebras task content and performance: A multinational study. *Informatics in Education*, 16, 39–59.
- Kibble, J.D., Johnson, T. (2011). Are faculty predictions or item taxonomies useful for estimating the outcome of multiple-choice examinations? *Advances in Physiology Education*, 35, 396–401.
- Kirschner, P.A. (2002). Cognitive load theory: implications of cognitive load theory on the design of learning. *Learning and Instruction*, 12, 1–10.
- Lee, F.-H., Heyworth, R. (2000). Problem complexity: a measure of problem difficulty in algebra by using computer. *Educational Journal*, 28(1), 85–107
- Leong, S.C. (2006). On varying the difficulty of test items. Paper presented at the 32nd Annual Conference of the International Association for Educational Assessment, Singapore.
- Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A. (2017). How presentation affects the difficulty of computational thinking tasks: an IRT analysis. In: *Proceedings: 17th Koli Calling Conference on Computing Education Research: Koli Calling 2017: November 16–19, 2017: Koli, Finland*. ACM, 60–69.
- Lonati, V., Monga, M., Morpurgo, A., Malchiodi, D., Calcagni, A. (2017). Promoting computational thinking skills: would you use this Bebras task?, In: *Proceedings of the International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP2017)*. Helsinki, Finland
- Lumley, T., Routitsky, A., Mendelovits, J., Ramalingam, A. (2012). A framework for predicting item difficulty in reading tests. ACEReSearch.
- Merrill, M.D. (1971). Necessary psychological conditions for defining instructional outcomes. *Educational Technology*, 11(8), 34–39.
- Newman, D.L., Kundert, D.K., Laner, D.S., Bull, K.S. (1988). Effect on varying item order on multiple-choice questions: Importance of statistical and cognitive difficulty. *Applied Measurement in education*, 1(1), 89–97.
- Schaffer, D., Doube, W., Tuovinen, J. (2003). Applying cognitive load theory to computer science education. In: M. Petre and D. Budgen (Eds.) *Proc. Joint Conf. EASE & PPIG*. 333–346.
- Shibli, D., West, R. (2018). Cognitive load theory and its application in the classroom. *Impact, Journal of the Chartered College of Teaching*.
- Van der Vegt, W. (2013). Predicting the difficulty level of a Bebras task. *Olympiads in Informatics*, 7, 132–139.
- Vora, K., Jain, S., Mehta, P., Sankhe, S. (2016). Predictive analysis: Assigning weightage and difficulty level of question using data mining. *International Journal of Computer Applications*, 138(9), 31–33.



W. van der Vegt is teacher's trainer in mathematics and computer science at Windesheim University for Applied Sciences in Zwolle, the Netherlands. He is one of the organizers of the Dutch Olympiad in Informatics and he joined the International Olympiad in Informatics since 1992. He is a part of the international Bebras community from the start in 2005 and is nowadays a member of the Bebras board, with a specific interest in task development.

REPORTS

Fostering Informatics Education through Teams Olympiad

Nadia AMAROLI¹, Giorgio AUDRITO², Luigi LAURA³

¹*IIS Aldini Valeriani Sirani, Bologna, Italy*

²*Department of Computer Science, University of Torino, Italy*

³*Department of Computer, Control, and Management Engineering,*

“Sapienza” University of Roma, Italy

e-mail: nadia.amaroli@istruzione.it, giorgio.audrito@unito.it, laura@dis.uniroma1.it

Abstract. Even though the International Olympiad in Informatics directly involves a restricted number of pupils from each country, one of its primary goals is stimulating interest in computer science and information technology over the whole younger segment of the world’s population. In several countries, this aim has to be accomplished without an active intervention of the Ministry of Education on school programs, relying on the efforts of small devoted organizations. In this context, promoting the involvement of a large number of school teachers may be as crucial as difficult to achieve. Following a 9-year experience of teams competitions in Italy, recently shared with other European countries, we argue that teams Olympiad may be an effective tool for widening the participation of high-school students and teachers, synergistically cooperating with existing individual competitions. On the one hand, teams contests foster peer education, encouraging talented students to help training fellows. On the other hand, these competitions can be more appealing both for average students, valuing group membership more than personal accomplishments, and most importantly, teachers: team achievements are more recognizably linked with the overall school or teacher performance than solitary excellences, resulting in increased returns rewarding the involved subjects.

Keywords: team work, programming contest, olympiads in informatics, peer education, programming training.

1. Introduction

Programming competitions are an effective tool to motivate and engage students in informatics concepts (Dagienė and Futschek, 2010). As such, the role played in the last thirty years by the two arguably most important competitions, i.e., the International Olympiads in Informatics (IOI) and the ACM International Collegiate Programming Contest (ICPC), is widely recognized (Audrito *et al.*, 2012; Bloomfield and Sotomayor, 2016; Combéfis *et al.*, 2017; Combéfis and Wautelet, 2014; Dagienė, 2010; Pavlova and Yanova, 2017; Tsvetkova, 2010).

In this paper we report our experience about the International Informatics Olympiad in Teams (IIOT), a recent competition aimed at filling an important gap in secondary school education: competing as a team of contestants (as in the university-targeted ACM ICPC) as opposed to the individual competition characteristic of the IOI. The IIOT builds over a 9-year experience of teams competitions in Italy and, in its second edition (2018), sees the participation of Italy, Romania, Russia, Bulgaria, Moldavia and Sweden.

During these few years of existence, the IIOT encouraged computational thinking and enhanced individual performances in Italy in a measurable way, confirming generally consensus on teamwork-based educational experiences (Tsay and Brady, 2012). In the specific contest of informatics education, teams contests proved to have a number of recognizable positive effects, allowing for peer education, sense of belonging, connection of talented students, involvement of teachers. The recent worldwide¹ success of the Bebras contest (Dagienė, 2008; Dagienė, 2010), aimed at promoting Informatics (Computer Science, or Computing) and computational thinking among school students at all ages, can also be seen as a proof of the effectiveness of team contests.

This paper is organized as follows:

- Section 2 details the IIOT characteristics and features, including its history, the competition and organizational structures, and, most importantly, how to join it.
- In Section 3 we discuss issues related to the diffusion of programming contests and their application to the IIOT.
- In Section 4 we report data from a national (Italian) point of view on the effects of teams contests.
- Concluding remarks are addressed in Section 5.

2. International Informatics Olympiad in Teams (IIOT)

The IIOT aims at motivating secondary school students to cultivate interest in informatics and coding, while testing and proving their problem solving skills, synergistically alongside the individual Olympiad in Informatics. Furthermore, the teamwork-oriented methodology used by this competition targets the exchanging of knowledge and experi-

¹ Currently, in the Bebras website www.bebras.org are listed 58 participating nations plus six that are planning to install a Bebras contest.

ence among young people with similar interests and qualifications, through the establishment of personal contacts both within and across countries.

More and more often, the world of work operates in contexts in which working groups are pivotal to carry out projects or activities. Teamwork skills, therefore, are usually a prerequisite for all those who are joining the current labor market: in all workplaces, or nearly so, you need to interface with other people to carry on your own activities. It is thus clear that establishing this capability can improve the performance level and lead to a better work environment.

In this competition, teamwork allows to achieve better results from the team's collective talents, from the members' ability to support each other in difficult times, and from the multiplied creativity that comes from the comparison of ideas. Ultimately, this competition experience translates into personal growth as well as improved individual performances, as we will detail in Section 4. Furthermore, the IIOT fills an age gap in team competitions between the Bebras and the ACM ICPC, being aimed at students in secondary schools.

Section 2.1 summarizes the competition history, while Sections 2.2 and 2.3 present the contest structure and organization respectively. Finally, Section 2.4 details on the possible ways to join the competition.

2.1. History of the Competition

Teams competitions in Italy started in 2010 with the *Olimpiadi di Informatica a Squadre* (OIS) thanks to Giorgeliana Carletto, who had the initial idea and the tireless perseverance to make this dream become a reality. In the first edition, participating teams belonged only to Emilia-Romagna (a region in the north of Italy, where the school of Giorgeliana Carletto is located), they were only seven, and the competition itself was carried out through the Croatian Open Competition in Informatics (COCI)². Nowadays, all of the twenty Italian regions participate into the contest, for a total of more than 400 teams, 2250 athletes and 110 schools involved in the 9th edition.

Starting from school year 2014/15 the contest is held on specifically-designed tasks, which were first written in Italian and later in English (from school year 2016/17) for two main reasons: to promote the international dimension of this competition, and because of the central role of this language in the today scientific community (and computer science in particular).

This first edition of the International Informatics Olympiad in Teams was held in Italy from May 17 to 20, 2017 at the IIS Aldini Valeriani-Sirani of Bologna, with the relevant award ceremony on 20 May 2017 at the presence of the former president of the European Commission and Italian Prime Minister Romano Prodi.

The second edition was held in Piatra Neam, Romania, from May 23 to 28, 2018. A mainly informative web site was built by Italian organization for the International project (<http://iio.team>), as well as a Facebook page of the event (<https://>

² <http://hsin.hr/coci>

fb.me/iio.team). There is also a website for each national competition: the Italian one is reachable at <http://oisquadre.it>, whilst the Romanian one is available at <http://cni.nt.edu.ro/ioit>.

Several personalities and public organizations have supported the project in Italy so far, including AICA, the Olympic Committee of the *Olimpiadi Italiane di Informatica* (OII), the former president of the European Commission and Italian Prime Minister Romano Prodi, Councilor Patrizio Bianchi of the Emilia-Romagna region, Ing. Romano Volta from Datalogic.

2.2. Competition Structure

Following the corresponding rule of the IOI, contestants are students enrolled in a school for secondary education, in the country they are representing, during at least September–December in the year before IIOT and who are not older than 20 on July 1st of the year of IIOT. Students who are studying abroad may represent the country of their nationality. Each team has to consist of four members and up to two reserves. A team can include at most one awarded contestant of the National Individual Olympiad in Informatics in the previous year. No student exchanges are allowed between teams. Each team is given two PCs, without Internet connection besides the official platform, no translators nor dictionaries.

Each national championship has its own rules that can be different from country to country. Usually, the national championship consists of four preliminary competition rounds and one national final round: one preliminary round per month starting from October till January, with the national finals held in the first half of March. Each team is given a username valid for the whole competition and a different password for each round. Each contest lasts 3 hours (4 for international rounds) and usually consists of 7 problems in English, to be solved in C, C++ or Pascal (other languages such as Java or Python may also be allowed in national contests) without special hardware or software requirements. Contestants may submit written questions through the platform to the Scientific Committee concerning the formulation and interpretation of the problems during each round.

The preliminary rounds are held online, on a national dedicated platform with an automatic evaluator (usually CMS (Maggiolo and Mascellani, 2012; Maggiolo *et al.*, 2014)). The national final competition, on-site, selects its participants according to the total scores obtained in the previous 4 rounds, but its score is not added to the previous ones. In Romania, the ten best-ranked teams are selected for the national finals. In Italy, finalist teams are selected based on their region: each region participates with its best-ranked team, plus the ten best teams nation-wise among the remaining ones.

The first one (or two³) winning teams of each national final competition will participate in the IIOT. All the involved countries can also participate with one more “special

³ The host country decides the number of participating teams, depending on the number of participating countries.

guest team”, from the national leader schools: their results will not be listed in the official rank, but they will be awarded.

The reference syllabus of the competitions is that of the IOI, however, most problems require a reduced competence set, such as: arrays, sorting and searching, greedy algorithms, recursion, dynamic programming, trees and basic graph algorithms. Given the limited time available to solve many problems, teamwork and cooperation are necessary for winning: no single individual can outperform a group of four people, working in pairs at a same problem and helping each other on the hardest tasks.

2.3. *Organizational Structure*

Enrollment in the national IIOT project (and in the IIOT itself) is completely free for all schools, both public and private. A school for each country is appointed as the *leader school* to coordinate all the activities of registration, competitions, administration of problems and awarding of the national winning team, as well as to maintain relationships with leading schools of other countries.

Each country in the IIOT is then represented by a National Committee (NC) that consists of four people: the representative of the Ministry of Education or another appropriate institution, the headmaster and a teacher of the leader school and the scientific coordinator. Together, the national committees of regular members form the International Committee (IC) and are responsible of regulation updates, overall organization, and appointing the International Coordinator who represents the IIOT project.

Meanwhile, the Scientific Committee (SC) of each regular participant country consists of few scientific experts, and has the duty to supervise the preparation and evaluation of tasks for the national contests. The host country SC should prepare tasks for the IIOT together with at least one extra proposal. These tasks will then be presented before the contest to the International Scientific Committee (ISC) consisting of the SC of each member country. The ISC has the right to reject tasks proposed by the host country SC, in which case the extra proposals will be considered as replacements.

The General Assembly (GA) is composed of the national committees of each participating country together with leaders of each participating team⁴ and a president nominated by the host country. The GA will approve the cutoff scores for gold, silver and bronze medals.

2.4. *How to Join the IIOT*

There are few different ways to approach to the IIOT, with increasing degrees of involvement. Firstly, you can unofficially attend contests as a team or individually, to test your ability or get to know the competition: starting from the last year, we host an online

⁴ Guest countries and teams are represented in the GA (even though not in the IC and SC). The distinction between guest and member countries will be clarified in Section 2.4.

mirror contest for this purpose. You can subscribe to the Italian training platform (Di Luigi *et al.*, 2016) before the contests start at <https://training.olinfo.it>, and during each round you can solve the problems by submitting your solutions. The official and mirror contests have identical tasks, rules and platforms, with the sole exception of starting times: the mirror contest is held USACO-style, so that you can choose when to start your 3-hour time window during the 24 hours following the start of the official contest. Unofficial ranking will be available as soon as the contest starts, on a separate web page than the official one.

Secondly, few schools from a country might want to experimentally join the official competition, testing themselves and their teams, and trying what attending this project means. As experimented this year with Sweden, we propose to freely host these teams (up to 10 for each country) on the Italian or Romanian official platforms, allowing the winning teams of each country to possibly participate as guest at the following IIOT. In this way, Sweden has held its national finals using the Romanian platform, during its contest timing.

Lastly, you might join the IIOT project officially by hosting your own national competition. In this case, you should contact us⁵ to organize your participation, set up an online contest management system such as CMS (Maggiolo and Mascellani, 2012; Maggiolo *et al.*, 2014), and the annual host country Scientific Committee will provide you their sets of problems for each of the qualifying contests, which you will be free to use, adapt or discard.

Joining the international project is free of charge and you can choose between two types of collaboration. First, you can be a *regular* member by giving a declaration of intent on hosting a future IIOT edition. In this case, your National Committee shall sit at the table as peer with other National Committees, and your participation into the International Final will be free of charge (except for travel expenses). Otherwise, you could attend it as *guest* participant: in this case, you will not need to host future IIOT editions nor take part into the decision tables, while still be allowed to attend the International final with your national teams, upon paying a participation fee decided upon discretion of the host country.

3. Organizing a Team-Based Informatics Contest

Successfully organizing a competition is a demanding task, involving countless choices for which a definitive answer may not even be possible. We hereby try to motivate and discuss the issues and solutions we encountered during these few years of experience, hoping they could provide a solid ground for further development and help newly joining countries for the setup of their national competitions.

Section 3.1 discusses the goals that motivated our quest for a better competition, Section 3.2 presents the technical issues encountered and respective solutions adopted, Section 3.3 lists the main organizational issues and solutions.

⁵ iiot@io.team

3.1. Motivating Goals

During our Italian experience, the basic motivating goal has always been widening the participation to the highest possible extent, in order to maximize the impact of the project. Towards this aim, we need to cope with the plurality and diversity of subjects that take different roles in the competition. Firstly, students may come from very different schools, with a theoretic or technical focus, and have varying sets of skills and competences. Teachers as well may have various backgrounds: mathematics (strongly represented in Italy), computer science or even electronics. Some of them, which are not acquainted with the practice of information technology, need to be supported by technicians in their school and/or possibly by the organizers themselves. University teachers and students need to be involved into the project as well: to prepare tasks, manage the contest platform, help teachers and technicians during the lab's preparation, and overall helping with the organization of the events.

In order for a competition to be successful, a broad participation must be achieved among all roles, by building a network linking together people united by their similar interests in informatics and problem solving. This bring us to the second goal: providing a fertile environment for building connections between different people joined by their passion, in this way encouraging their positive involvement in the subject.

Meanwhile, we are obviously interested into our students personal growth. For a team-based contest, this process can simultaneously involve social and teamwork skills on one side, technical coding and problem solving skills on the other side. As our milestone goal, we aim to improve the performance of students in individual competitions and coding in general, while encouraging a cooperative attitude instead of a more solitary or "nerd" one: looking at mates with respect and tolerance, not estranged but rather actively immersed into the surrounding environment. Regarding the technical skills, we aim in particular at providing a fruitful training for the individual Olympiad: as contests are held regularly and progressively, students can be lead through a growth path directed at their personal performance in individual contests, learning new skills from teachers, mates and experience.

3.2. Technical and Scientific Issues

Several technical and scientific issues have to be addressed in order to met the goals just stated. Firstly, in order to handle online contests with thousands of students simultaneously competing, we had to host the CMS platform on the Google Cloud Platform as locally available servers were failing to deliver an acceptable quality of service. We used a standard machine with 50GB SSD storage for the database and main instance of CMS, together with three preemptible machines devoted to the Contest Web Server, and eight preemptible worker machines with high CPU settings. Virtual worker machines on the cloud proved to have higher variability in execution times than worker processes on

locally controlled hardware, however, the difference was not so significant to impair the evaluation process.

Another subtle issue with online contests is that of ensuring a fair competition without any form of physical control: to address this issue, we opted for an approach mixing automatic tools and teacher accountability. We handed multi-platform instructions to teachers on how to block Internet access for contest computers, while implementing an automatic connection check in the contest website, which sends alert mails to teachers whenever one of their teams is spotted to be connected to the Internet. After the end of the contest, teachers are queried about their teams who hit a certain threshold number of warnings (if any), and we decide together with them whether to penalize the teams in question. Furthermore, a semi-automatic check for plagiarisms is done based on JPlag,⁶ and plagiarisms found are correspondingly penalized. Finally, we plan to implement in CMS for next year a checker ensuring that submitted source code does not contain *pragma* instructions and *inline assembly*, which are against the spirit of the competition and prohibited in other contests such as the ACM ICPC.

The selection and preparation of tasks requires a similarly careful attention than that of the technical framework. In order to engage students with different preparation levels and back-grounds, we attempt to build set of tasks with amusing narratives and widely varying scopes, ranging from extremely easy to technically involved or logically tricky. This requires contests made from several problems: we believe that the chosen number of seven problems is appropriate for this purpose. Even though this number is lower than the average number of problems in the ACM ICPC team contests, the discrepancy is mitigated by the presence of subtasks of various difficulty into which each task is fragmented. For next year, we plan to encourage the participation of younger and less experienced contestants by introducing a progressive syllabus, dividing required competences into three increasing levels: a starting level for newbie programmers, an intermediate level with selected topics from school programs, and an advanced level corresponding to the IOI syllabus. The tasks will then be sorted and explicitly marked according to their syllabus level.

As previously mentioned, the involvement of teachers is crucial for the correct running of the competitions, and even more so for enriching the participation and preparation of students. Since team performances are more easily connected to quality teaching than solitary excellences, teachers are often more motivated to get involved into team contests than into individual competitions. In order to promote the involvement of teachers into the competition, we plan for next year to allow and encourage teachers to submit tasks for consideration into the competitions, while acknowledging the task author in the final competition booklets.

3.3. Organizational Issues

Among the organizational issues, the first one we had to face is schools and teams enrollment. In the first editions of the OIS, we managed enrollment manually through for-

⁶ <https://jplag.ipd.kit.edu>

mal emails. As the number of participating schools and teams grew, manual enrollment became unfeasible, and we had to switch to an online registration form: a first online form is designed for enrolling schools, and then a second online form is sent to enrolled schools for submitting teams. These forms are advertised to schools across all Italy with the help of the Ministry of Education.

Once the enrollment has taken place, it is crucial to keep active channels of communication between the organization and school referents. With the plethora of communication systems available in the modern world (phone calls, text messages, emails, WhatsApp or Telegram) it is sometimes difficult to reach an agreement on a preferred set of communication methods. At the moment, we are encouraging the usage of email for formal communications, and of WhatsApp for urgent, non-official messages. These channels are used for general information and support, but also to gather the school feedbacks needed to pursue the evolution of the competition. Several ideas arising from valuable feedback have been (or are going to be) implemented, such as: unofficial mirror contests, cooperative Internet access control, progressive syllabus.

Besides the communication problems, other issues arise in the practical organization of the national final and international contest. Firstly, computer labs with substantial capacity need to be found: about 60–70 PCs in the national round, 20–30 in the international one (currently). So far, we alleviated part of the technical burden by hosting online even the on-site competitions, reducing needs for specific hardware or configurations, and thus being able to host the competitions in sufficiently large schools (IIS Aldini Valeriani Sirani, Bologna).

Secondly, accommodation and cultural activities need to be prepared for the about 150 people (50 for the international contest) participating in the event. In order to increase the reach of the event, we also have to look for VIPs to attend the award ceremony, set up entertainment and a buffet, contact press agencies and media. All of this activities come at a cost, so that funding is usually the most critical issue. There is no magic recipe for fund-raising, and so far we obtained the most from long-term low-budget sponsoring from local businesses, integrated with limited institutional funding. For the upcoming years, we hope to score sponsorships from larger international companies as well (Google and/or Microsoft).

4. Impact of the Teams Olympiad in Italy

During these years of teams Olympiad in Italy we collected several positive feedbacks. First of all, we obtained positive reactions from students and teachers resulting in a so-far increasing number of schools and students participating in the teams Olympiad, shown in Fig. 1. The major spike in the graph corresponds to the year 2015 when the teams Olympiad were first disconnected from the COCI and popularized through all Italy. Despite this increasing trend, the number of schools and students participating is still much lower than that of the individual Olympiad: 2250 versus 14531 participants and 110 versus 565 schools in 2018 (below 20% in both cases). Thus, there is still leeway for a further future increase.

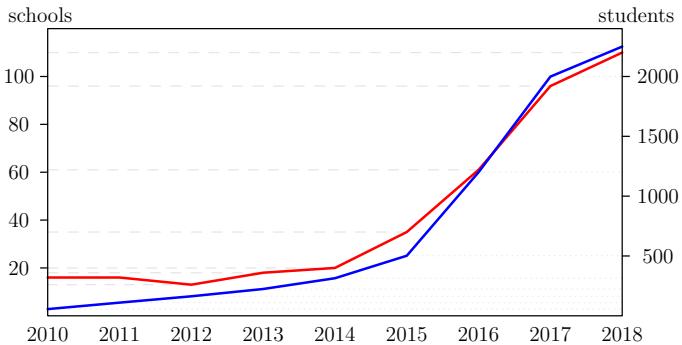


Fig. 1. Schools (red) and students (blue) participating in the Italian teams Olympiad.

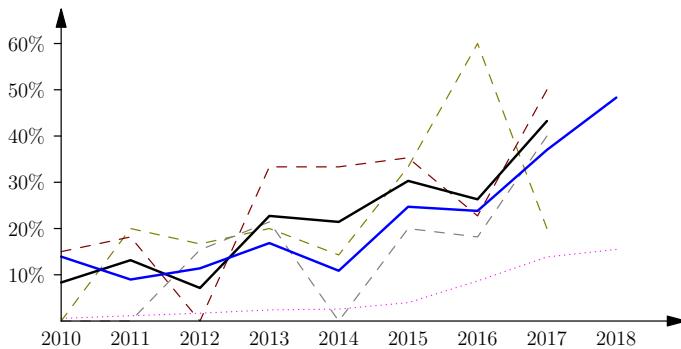


Fig. 2. Percentage of teams Olympiad contestants among individual contestants (dotted magenta), among national finalists of the individual Olympiad (blue), and among medalists in the national contest (dashed bronze, silver, gold, and solid black overall).

As trainers for the individual Olympiad, we also tried to test the effectiveness of the team Olympiads as a form of training and involvement into the subject. In fact, participation in the teams Olympiad highly correlates with better performances at the individual contests, as shown in Fig. 2: a small initial number of students (1% to 15% during the years⁷) scores a large part of the available places and medals at the individual Italian national contest (about 10% to 40% during the years), quite consistently among the different score levels (non medalists, bronze, silver and gold medals). This correlation is due to two cooperating factors: that better students are more eager to participate to new contests, and that students get better by participating into the teams Olympiad.

Among these factors, we claim that the second plays the more relevant role: students get better by practicing with the teams Olympiad. Two main observations substantiate this claim. Firstly, students first participate in the teams Olympiad and then

⁷ Data about 2018 medalists is not available as the individual finals are yet to happen at the time of this writing.

score medals in individual contests more often than medalists happen to join the teams Olympiad. In fact, the regulations forbade medalist to join teams until last year, so that all the data depicted in Fig. 2 regarded no medalists joining the teams Olympiad. Secondly, the overall level of the Italian individual competition is getting better over the last few years.

This increase in the competition level is not only acknowledged by trainers, but also visible in the international results of the Italian team at the IOI (see Fig. 3), where a consistent improvement has taken place in the last few years reaching an all-time best result in 2017 (as average quantile). As a further benchmark of the competition level, we compared the individual informatics Olympiad with the individual mathematics Olympiad. Since 2012, the best students in the Italian mathematical Olympiad are encouraged to join the informatics Olympiad with a special selection process, resulting in a few students being invited to the national contest. Assuming the level of the Italian mathematical Olympiad is approximately constant, we nonetheless observed that a decreasing number of people from this alternate selection is able to reach the training camps (see Fig. 4), suggesting that the level of the main selection is improving.

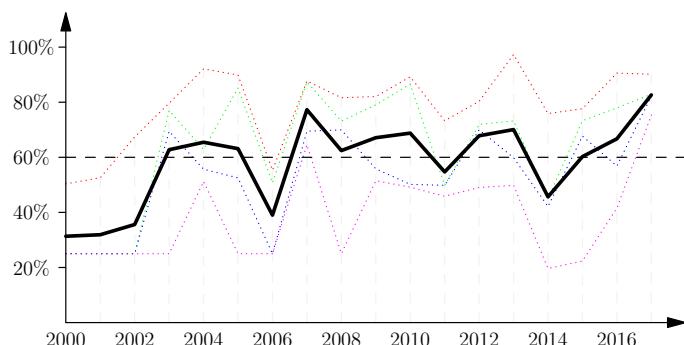


Fig. 3. Individual (dotted) and average (solid black) results of the Italian IOI teams as quantiles (i.e., position on scoreboard as percentage), as reported in <http://stats.ioinformatics.org>. Missing data for old non-medalists is approximated to the average 25% quantile. Performance of the four athletes is colored magenta, blue, green, red from 4th to 1st ranked respectively.

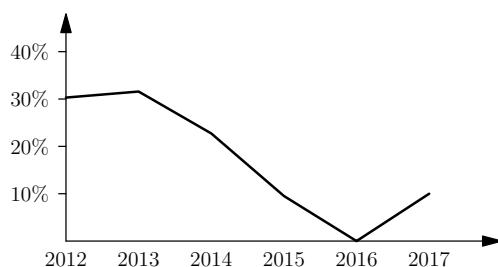


Fig. 4. Percentage of mathematical Olympiad contestants among Italian finalists.

As the effectiveness of teamwork in education is widely accepted and acknowledged (Tsay and Brady, 2012), we find these results unsurprising. In fact, the specific teamwork promoted by the teams Olympiad can have additional benefits than teamwork in a generic setting. In most Italian high schools, gifted students are evenly distributed across classes, so that they usually do not know each other. Teams Olympiad thus provide a unique setting connecting talented students in the same school, enabling not only peer education but also a sense of belonging that encourages students to pursue their passions and goals.

5. Conclusions

In conclusion, we claim that the Olympiad in Informatics in Teams, besides allowing students involved to learn essential Informatics, English and teamwork skills, can significantly improve the students' results for the IOI. In our experience, working together with a team and comparing your own ideas with mates can improve critical thinking, suggest new approaches and solution methods, and promote passion for the subject. Moreover, during its four online contests before the national one, the teams Olympiad helps students during their preparation for the individual Olympiad through problems with bit by bit growing difficulties, engaging them with easier problems and gradually driving them to more complex topics. These claims are corroborated by data on the Italian competitions in the last decade, which shows not only a correlation between participation in teams and overall results, but also an overall positive impact on the competition level. As a consequence, a future expansion of this competition to more countries might be able to positively effect the IOI in a significant way. Towards this aim, we propose four ways to join the teams competition: individually through the mirror online contests, experimentally by relying on the Italian or Romanian organization for the technical side, and regularly both as guest (non-voting, non-hosting) or as member country. We hope these parallel paths will help in engaging an increasing number of students and countries, by allowing anyone to join according to the level of involvement he is able to ensure.

Acknowledgements

We thank Giorgeliana Carletto for starting both the Italian Informatics Olympiad in Teams (OIS) first and IIOT later, but mostly for providing us an example of what a single individual can achieve when supported by passion and devotion for a greater goal. We also thank all the teachers that believed in the IIOT project, both at a national and international scale, and helped us to spread the diffusion of computer science and computational thinking. Last but not least, we are thankful to Gabriele Farina, Edoardo Morassutto, Luca Chiodini, and William Di Luigi, that managed to make everything work during the contests.

References

- Audrito, G., Demo, G.B., Giovannetti, E. (2012). The role of contests in changing informatics education: A local view. *Olympiads in Informatics*, 6.
- Bloomfield, A., Sotomayor, B. (2016). A programming contest strategy guide. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE'16*. New York, NY, USA. ACM, 609–614.
- Combéfis, S., Barry, S.A., Crappe, M., David, M., de Moffarts, G., Hachez, H., Kessels, J. (2017). Learning and teaching algorithm design and optimisation using contests tasks. *Olympiads in Informatics*, 11.
- Combéfis, S., Wautellet, J. (2014). Programming trainings and informatics teaching through online contests. *Olympiads in Informatics*, 8.
- Dagienė, V. (2008). The bebras contest on informatics and computer literacy-students drive to science education. In: *Joint Open and Working IFIP Conference. ICT and Learning for the Net Generation, Kuala Lumpur*. 214–223.
- Dagienė, V. (2010). Sustaining informatics education by contests. In: *International Conference on Informatics in Secondary Schools-Evolution and Perspectives*. Springer, 1–12.
- Dagienė, V., Futschek, G. (2010). Introducing informatics concepts through a contest. In: *IFIP Working Conference: New Developments in ICT and Education. Universite de Picardie Jules Verne, Amiens*.
- Di Luigi, W., Farina, G., Laura, L., Nanni, U., Temperini, M., Versari, L. (2016). oii-web: an interactive online programming contest training system. *Olympiads in Informatics*, 10, 195–205.
- Maggioli, S., Mascellani, G. (2012). Introducing cms: a contest management system. *Olympiads in Informatics*, 6, 86–99.
- Maggioli, S., Mascellani, G., Wehrstedt, L. (2014). Cms: a growing grading system. *Olympiads in Informatics*, 123.
- Pavlova, O., Yanova, E. (2017). Olympiads in informatics as a mechanism of training world-class professionals in ict. *Olympiads in Informatics*, 11.
- Tsay, M., Brady, M. (2012). A case study of cooperative learning and communication pedagogy: Does working in teams make a difference? *Journal of the Scholarship of Teaching and Learning*, 10(2), 78–89.
- Tsvetkova, M.S. (2010). The olympiads in informatics as a part of the state program of school informatization in russia. *Olympiads in Informatics*, 4.



N. Amaroli is a secondary school teacher in computer science labs and a regional school office trainer for digital competences and teaching with innovation technologies. She is the International Coordinator of the IIOT and the deputy president of the OIS. She is the author of the computer science section in the online interactive course for secondary school students, organized by Emilia-Romagna regional school office and by Italian Ministry of Education.



G. Audrito is involved in the training of the Italian team for the IOI since 2006, and since 2013 is the team leader of the Italian team. Since 2014 he has been coordinating the scientific preparation of the OIS and of the first edition of the IIOT. He got a Ph.D. in Mathematics in the University of Turin, and currently teaches “Object Oriented Programming” in the “Piemonte Orientale” University and works as research assistant in the University of Turin.



L. Laura is involved in the training of the Italian team for the IOI since 2007, and since 2012 is in the organizing committee of the Italian Olympiads in Informatics. He got a Ph.D. in Computer Science in the “Sapienza” University of Rome, and currently teaches “Web-based Systems Design” in the Tor Vergata University of Rome and “Information Systems” and “Machine Learning” in the LUISS University of Rome.

PRASK – an Algorithmic Competition for Middle Schoolers in Slovakia

Michal ANDERLE

*Faculty of Mathematics, Physics and Informatics of the Comenius University
Bratislava, Slovakia,
e-mail: anderle.michal@gmail.com*

Abstract. Although informatics education is compulsory in Slovak middle schools, its curriculum is insufficient to prepare the pupils for competitions such as the national Olympiad in informatics. There is a huge gap in knowledge that needs to be addressed. To reduce this problem, PRASK, an algorithmic contest for middle schoolers (approx. ages 10–15), was created in 2015. In this paper we discuss the main concepts behind this competition, ways of dealing with insufficient computational knowledge, types of tasks used to engage and educate young competitors, and some results and lessons learned from the first four years of this competition.

Keywords: algorithmic competition, middle schools, PRASK.

Introduction

Computer science is a rapidly evolving discipline, therefore schools are struggling to reflect this progress in their education process. In addition to this fact, computer science is mostly targeted at high school students, and curricula for lower grades has been emerging only recently.

From the very beginning, education of computer science for gifted pupils was taken over by universities, various competitions and volunteer organizations. This trend has been present in Slovakia for more than thirty years. However, the largest algorithmic competitions in Slovakia focuses only on high school students.

This brings up the question whether a similar format of competitions could be used in middle schools. In this article we will introduce such competition, PRASK, point out changes that were necessary to incorporate in order to adapt the competition to a new audience and also problems that occurred.

Algorithmic Competition PRASK

History of Correspondence Seminars

Dagienė and Futschek (2010) believe that many countries are lacking high-quality computer science teachers who would be able to introduce the pupils to the computer science in an interesting way. It is therefore becoming more and more common that instead of developing algorithmic skills, the use of specific software applications is trained.

However, many authors believe (Dagienė and Futschek, 2010; Dagienė *et al.*, 2015; Dagienė and Stupurienė, 2016; Forišek and Winczer, 2006; Kalas and Tomcsanyiova, 2009; Kubica and Radoszewski, 2010) that by means of informatics competitions, we can present various parts and used concepts of computer science. Competitions like Olympiad in informatics, Bebras and many others have been the result of this line of thinking. And also many years of experience show that such a way of presenting computer science can be very effective.

Slovak Republic has a rich history of algorithmic competitions. The Olympiad in informatics was established in 1985 and has been educating talented pupils for more than 33 years. There is even older competition, Correspondence seminar in programming (KSP), created in 1983. This competition serves as a stepping stone for all pupils interested in computer science, primarily high school ones.

Correspondence seminars organized for high school students have a long tradition in Slovakia and the Czech Republic, not only in computer science but also in mathematics and physics. The original purpose of the seminars was to educate talented pupils in the natural sciences. Another reason for its creation was also the above-mentioned shortage of qualified teachers. Unfortunately, this has not changed in 35 years (Forišek, 2007), and this form of non-formal education remains key in teaching of gifted pupils. Even seminars for middle schools have been appearing.

Origin of the PRASK Competition

There were multiple reasons for creating algorithmic competition for middle schoolers and they all came together in 2015. Perhaps the most significant one was the feeling of the original founders that pupils are getting to algorithmization and programming relatively late, if at all. This negatively affected competitions like KSP and Olympiad, which lacked young competitors.

It is important to note, the pupils' interest in algorithmization and programming did exist, and it has also seemed that basic algorithmic concepts could be taught even before high school. KSP organized multiple programming schools where these pre-

sumptions were confirmed. Moreover, for mathematics and physics, there were already successful equivalents of high school seminars for middle schoolers, so the PRASK competition was created.

First of all, objectives of this competition and the basic assumptions had to be clarified. PRASK was intended for talented pupils of middle schools interested in computer science. However, the middle school in Slovakia consists of five years (10–15 years old) in which the knowledge of pupils changes considerably. Consequently, the competition primarily focuses on the last three years of middle school. But of course, involvement of the younger pupils is always welcomed. This restriction determines the degree of mathematical knowledge that is to be expected from all competitors, based on the Slovak national curriculum.

However, organizers do not assume any common knowledge in field of computer science. From personal experience and interviews with pupils, it was clear that different schools taught different things and used different environments and tools. Therefore, the organizers had to assume that the prior knowledge of pupils in computer science will largely vary.

The main goal of the PRASK competition was promotion and development of algorithmic and programming skills. The contest was meant to reach out to a complete beginners who had not came across programming and algorithms whatsoever, but at the same time provide considerable challenge to more experienced competitors. That should provide scaffolding to the high school competitions like KSP and Olympiad in informatics.

Format of the Competition

The format of the PRASK competition was strongly inspired by KSP. Annually, two parts of the competition are held, each consisting of two rounds. Within one round, five tasks are published. There are no performance categories, the tasks are assigned for everyone. Each round has a set deadline for the pupils to submit their solutions. There is at least one month between the release of the tasks and the final deadline.

Each task has its own means of solution – some require uploading a program, other written description of the solution. After the deadline, the pupils' solutions are graded and the feedback is sent back in the form of a comment. Feedback contains commendation for good work, explanation of mistakes and grading, and some follow-up questions to the task.

At the end of both parts, the week long camp is organized for top 18 competitors. These camps consist of algorithmic lectures, but also sports, games and team building activities. Program of the camp is focused on presenting computer science as a fun and interesting topic, and on building a community of young people interested in it. This community building aspect is very important, because it ensures continuity (Forišek and Winczer, 2006).

Tasks

The PRASK contains three types of tasks – theoretical, practical and programming. Each type covers different area of informatics and ultimately helps developing different abilities. Each round contains two theoretical, two programming and one practical task.

Programming Tasks

One of the main goals of the PRASK is to develop pupils' programming skills. Therefore, each round contains two programming tasks. In these tasks, competitors need to write and debug a program that solves assigned task. The uploaded solutions are automatically tested and the competitor finds out his/her results immediately. The solutions can be then reworked and submitted again until the deadline.

The tasks often contain multiple easier subtasks worth partial points and the whole process is similar to the one at the IOI. Solutions can be written in several programming languages, Python and C++ are most frequently used.

The difficulty of programming tasks is determined by a model solution. It should only use basic concepts – variables, cycles, conditions and arrays. In harder tasks, it is also possible to use algorithms and data structures that are implemented in the used languages, most commonly sorting and binary search trees. Model solutions should not require knowledge of any advanced algorithms or data structures, which usually excludes graphs. The recursion is also avoided.

However, considering these limitations, solutions require some non-trivial idea that leads to a more effective solution. These tasks should not be just straightforward implementations. But as mentioned above, subtasks are commonly used, meaning that trivial or slower solutions always score at least some points.

It is clear that this type of task would be inappropriate for beginners. PRASK should be accessible to all pupils interested in computer science regardless of whether they know how to use a programming language or not. Hence, the programming Hatchery was created. It consists of four sets of tasks and study texts that create a tutorial to C++. In these four sets, pupils learn to use variables, conditions, cycles and arrays. And because this competition would require much more effort from beginners (first learn programming language in Hatchery and then solve problems), points obtained in the Hatchery are able to replace points from programming tasks. Each set of tasks in Hatchery can replace one task in PRASK, which means that beginner contestant can learn programming in half a year and then use this new knowledge to later solve tasks.

Finally, we add that the choice of C++ is purely practical. Besides the fact, that organizers themselves have more experience with this programming language, and they were able to use existing materials when creating the Hatchery, the C++ have some advantages over Python, e.g. support at IOI and Slovak national competitions, due to its speed.

Practical Tasks

Practical tasks are often interactive and present new technology or part of computer science to the pupils. Their main goal is to promote IT and to motivate pupils to further work. The interactivity and the unusualness of these tasks is an attraction for contestants and indeed, this type of task is the most popular one. Presented problem is often a puzzle, that must be solved and the feedback is immediate.

A nice example of such task is the very first task, in which pupils were referred to a purely black web page containing a secret password. Pupils had to figure out that page contains images of black letters, so the background of the page needs to change. It was up to them to use Javascript, view the source code of the page, edit it locally or use entirely different approach.

During the four years of this competition, pupils had to use Google, Excel, Word or various image editors in new, inventive ways. Practical tasks also include problems, that need to be solved by using specific technology, for example AutoHotKey for automatic mouse and keyboard control, SQL or Prolog. For these problems, a quick tutorial is presented, containing the necessary concepts and commands. Contestants need to learn how to work with them and use them effectively. To help them do that, interactive web environments and easy subproblems are offered.

Even though the tasks might be at times technologically challenging, it seems that even such tasks do not discourage contestants. In the survey, that was performed in January 2018, nearly 93% of participants stated that they had little to no technical problems while competing in PRASK. This significant percentage is probably achieved by the existence of tutorials that are applicable to different systems (Windows, Linux, Mac), as well as the involvement of the organizers who can be contacted by the pupils at any time.

Theoretical Tasks

Probably the most unconventional and challenging to prepare are theoretical tasks. These tasks direct contestants to design some of the known algorithms or dive deep into specific area of computer science. There were tasks based on well-known algorithms or data structures (spanning trees, binary search trees) but formal languages and automata are commonly used as well. Previous knowledge is not necessary, organizers even assume that presented problems are not known by competitors, task statement contains all important rules and relationships, contestant needs to combine them, come up with their new usage and formalize their thinking.

For this type of problem, participants submit a text document describing their solution. Description needs to contain not just an answer, but also approach they used to get the answer. Some sort of formalized “algorithm” is often required as well, either in pseudo-code or natural language.

Theoretical tasks are probably most different from practises used at IOI but are actually very reminiscent to the format of Slovak national olympiad and KSP. It focuses on thinking process, probably the most important ability to develop. As Michal Forišek stated (Forišek, 2007) “practice only” competition style (such as IOI) are restricted by actual implementation and marginal issues such as debugging techniques, library knowledge, etc. Slovakia is a rather small country and the number of participants in this type of competition peaks at one hundred. Hence, very personal approach can be used.

In PRASK, this type of task lacks immediate feedback, solutions are corrected after the deadline. But benefits include more personalized and detailed feedback and focus on the thinking process of the contestant. It is not even uncommon to have different expectations on different contestants and score them adequately to it. Even though PRASK presents itself as a competition, the personal growth and education of participants is even more important.

Unfortunately, theoretical tasks are the least popular among the contestants. The main reason for this is the need to write down solutions. Up to 50% of contestants said that they were discouraged by it. This result is not surprising, write down solution takes most of the time and pupils are not used to it. On the other hand, we believe that this part of competition is beneficial for the pupils, improving their ability to express and write understandable procedures and algorithms.

Concepts used for theoretical task sometimes include one stroke drawing, various dynamic programmings, regular expressions, error detection and correction codes, logic gates, deterministic finite automata, mergesort, minimal spanning tree, Euclid's algorithm and others.

Experience and Challenges for the Future

Preparation of Theoretical Tasks

Because theoretical tasks are the most atypical and also the most interesting and challenging to prepare, we will discuss their preparation in more detail. As we mentioned before, theoretical tasks are used in the Slovak national olympiad as well as in correspondence seminar. However, they use is a bit differently. Most of the times they are classic tasks, but instead of implementation, contestants need to write down a description of solution. They rely on existing knowledge, often using classic algorithms and data structures. These tasks cannot be used in PRASK, where the tasks should be solvable from the scratch. This forces organizers to create entirely new, original tasks fulfilling all the assumptions that are put into them. In fact, we are not aware of any other competition that would be using similar tasks.

All of this put a lot of responsibility on tasks seters. In order for these tasks to be solvable and understandable, a suitable form must be used. Tasks cannot use technical terms, problem description uses a story that presents a problem in a more comprehen-

sible form. Similar stories are used at IOI, but their part in PRASK is more crucial. This story offers a metaphor with which pupils can work more easily. If they never encountered graph problem, we cannot use terms as vertex or edge, we need to present e.g. cities interconnected by roads. Note, that choice of a suitable metaphor is really important, helping pupils to work more efficiently with original abstract concepts and even leading them to the right solution (Forišek and Steinová, 2013). Additionally to metaphor, theoretical tasks tend to include images or interactive environments pointing to various special cases that may occur. That also helps pupils to better understand presented problem.

To create problems that are accessible for beginners and challenging for experienced contestants, theoretical tasks are divided into several, successively more complicated, subtasks. These subtasks are designed to progressively guide participants towards the solution. The first subtask presents specific inputs and contestants can solve them using just pen and paper. While solving these easy subtasks, pupil will develop some sort of strategy or algorithm. Next subtasks ask the pupil to formalize this algorithm and formulate it in natural language. The hardest subtasks, meant for the experienced contestants, often ask for a proof of correctness.

The last issue the PRASK contestants have to deal with is an estimate of time complexity. It is a basic principle that must be taken into consideration when designing an efficient algorithm and the means by which the solutions will be evaluated. Therefore, efficiency must appear in the statement in some form. Most of the time, intuitive view of complexity is used, asking from pupils to create algorithm that would be feasible even with pen and paper on larger inputs. From this description is obvious that backtracking all possibilities is correct but not fast enough. Alternatively, in some problems a number of specific steps can be used. For example, when sorting, pupils were told to minimize the number of comparisons.

Involvement of High School Contestant

The Correspondence seminar in programming is organized by university students at Comenius University and is targeted at high school students. Since the PRASK is intended for middle schoolers, high school students can be involved in its organization. In fact, half of the current organizers do not attend university yet.

The selected contestants of KSP are offered the opportunity to participate in the preparation of PRASK. This selection is based on knowledge of computer science, age and personal interest. These young organizers are preparing a vast part of competition, writing statements and solutions, preparing test data, and planning camp activities and lectures. University organizers serve as their mentors, check quality of the prepared materials and give feedback. Also tasks ideas originate mostly from senior organizers.

This cooperation is very interesting and fulfilling for both sides. Seniors have the opportunity to share their experience and high school students can improve in several different skills. On the one hand, their understanding of computer science arises. Being

able to prepare and present lecture or create test data for algorithmic problem needs deeper understanding of underlying concepts. On the other hand, in these activities are also involved important soft skills. Write a clear task statement, come up with a fun idea for camp game and implement it, and take responsibility for the participants are qualities, that all will be useful in their future lives.

It also helps with continuity. Many of these students will continue to be involved in similar activities during their university years. Either in PRASK, KSP or the Slovak national Olympiad. Helping to prepare them from young age is beneficial to all of these competitions.

Statistics

In this part, we will take a look at some data from the first three years of PRASK competition. First, let's look at the number of contestants in each round, shown in Fig. 1. It is important to note, that these numbers can be misleading. Rounds 1 and 2, as well as rounds 3 and 4, in each year belong to one part and they are scored together. Therefore, number of contestants in round 2 (or 4) is union of contestants in round 1 and 2 (3 and 4).

But, there is a part of contestants that only solved few easy subtasks of some interactive problem, but overall they got very few points. Therefore, in Fig. 2 we will show the number of contestants with at least half the points. We believe that it is possible to get at least half of the points with a bit of effort and these participants are therefore reasonably engaged in the competition.

Notice, that the number of contestants peaked during round 1 in 2015 and round 3 in 2016. In these two rounds, instead of programming tasks, the interactive and playful

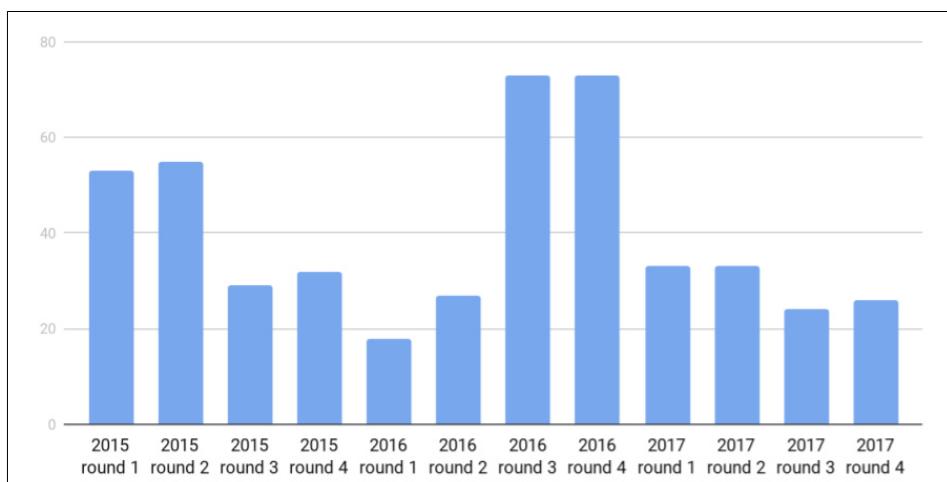


Fig. 1. Number of contestants in PRASK competition.

self-designed environments were used. Both of them were very popular, which resulted in a higher participation rate.

The average number of points for contestant is 35.58 out of 75 possible. However, if we take a look only at participants with at least half of the points, it grows to 56.44 which is more than 75%. In the Fig. 3, we can take a look at the average percentage for each tasks published during first three years. Notice, that almost all of them range between 50 and 75 percent. We can assume that the tasks are reasonably difficult and it is fairly easy to get substantial part of the points, mainly thanks to the easiest subtasks.

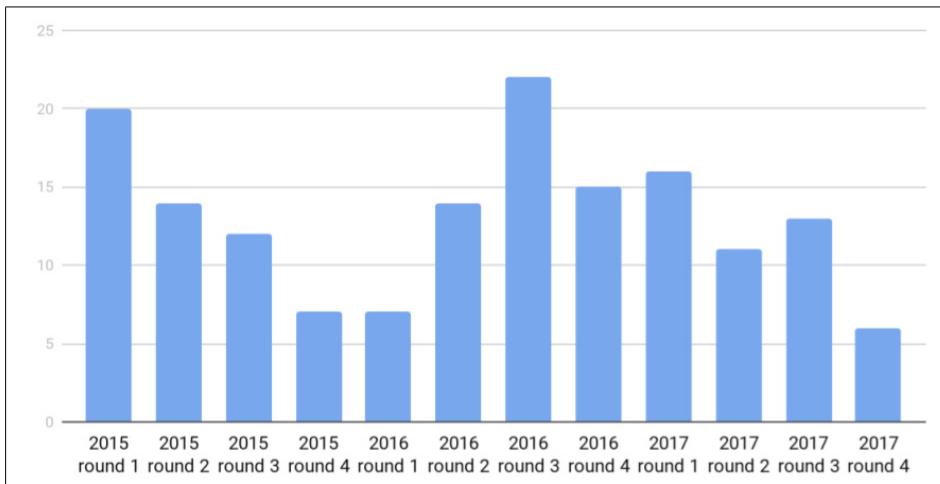


Fig. 2. Number of contestants with at least half the points.

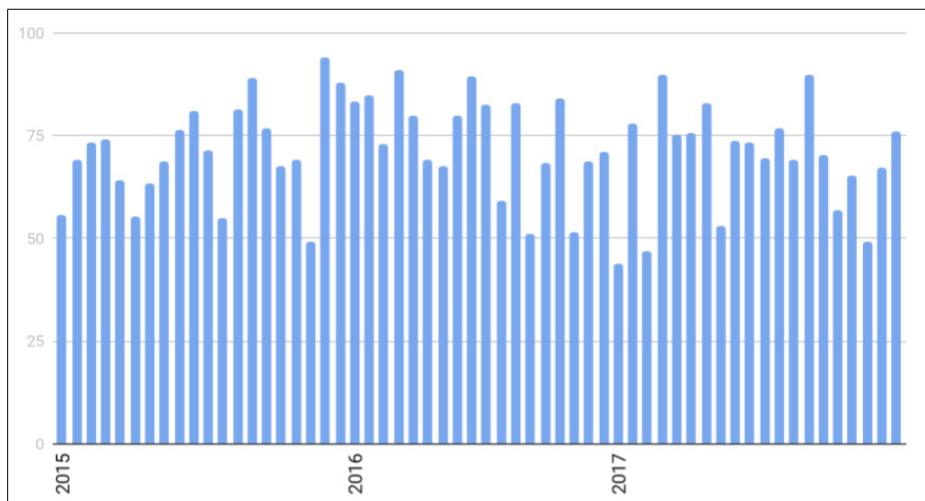


Fig. 3. Average percentage in individual tasks.

Finally, we will mention two interesting things that are present in scoreboards. The first is that girls took a rather dominating spot. Although the majority of contestants are still boys, first three positions during the three years were occupied only by girls. The second interesting point is that PRASK has had some very young competitors. There have always competed at least one fifth grader (age 10–11) and the youngest contestant was a girl from fourth grade (10 years old, not even in middle school).

Future Challenges

PRASK is now being organized for almost four years and the fifth year is being prepared. It is still evolving and there are several issues that need to be addressed.

Probably the most important is low participation count. On average, there are around thirty participants in each round, a number we would like to increase. Since PRASK is still fairly young, it is not well known among pupils or teachers. Teachers are the target category we would like to focus on at first, as they have great impact on huge number of students.

Other problem PRASK is facing is creation of consistent problem sets. Organizers are not sure about intended complexity of the tasks, which leads to trial and error approach. Also, it is really hard to prepare high quality theoretical tasks, which poses the biggest challenge in problem setting. Theoretical tasks also need to be popularized among contestants. Using interactive environments would be the best, as the survey showed, but those environments are time consuming to prepare, so the fine line needs to be found.

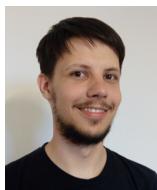
Conclusion

We presented PRASK, algorithmic competition for middle schoolers, and looked at the experience gained from the first four years of organizing it. This competition does not imply any prerequisites on previous algorithmic knowledge, trying to offer pupils an environment in which they can grow. The tasks are trying to concentrate on the idea of the solution rather than the actual implementation.

It turns out that pupils have fun dealing with interactive tasks in which they receive immediate feedback or tasks in the form of puzzles. In our opinion, theoretical tasks have tremendous potential but they are not as popular because of the need to write down description of the solution. For the future, the greatest challenge is to increase participation rate and improve pupils' interest in theoretical tasks.

References

- Dagienė, V., Futschek, G. (2010). Introducing informatics concepts through a contest. In: *IFIP Working Conference: New Developments in ICT and Education. Université de Picardie Jules Verne, Amiens, 2010.*
- Dagienė, V., Pelikis, E., Stupurienė, G. (2015). Introducing computational thinking through a contest on informatics: Problem-solving and gender issues. *Informacijos Mokslai/Information Sciences*, 73.
- Dagienė, V., Stupurienė, G. (2016). Informatics concepts and computational thinking in K-12 education: A Lithuanian perspective. *Journal of Information Processing*, 24(4):732–739.
- Forišek M. (2007). Slovak IOI 2007 team selection and preparation. *Olympiads in Informatics*, 1, 57–65.
- Forišek, M., Steinová, M. (2013). *Explaining Algorithms Using Metaphors*. Springer.
- Forišek, M., Winczer, M. (2006). Non-formal activities as scaffolding to informatics achievement. *Information Technologies at School*, 529–534, 2006.
- Kalas, I., Tomcsanyiova, M. (2009). Students' attitude to programming in modern informatics. In: *Proc. 9th WCCE 2009, Education and Technology for a Better World.*
- Kubica, M., Radoszewski, J. (2010). Algorithms without programming. *Olympiads in Informatics*, 4.



M. Anderle is a PhD student at the Comenius University in Slovakia. Since 2011, when he joined the university, he has been involved in the organization of various algorithmic competitions, mostly the Slovak national olympiad and Slovak correspondence seminar in programming, and summer schools. He is also a cofounder of the algorithmic competition for middle schoolers – PRASK.

Grading Systems for Algorithmic Contests

Ágnes ERDŐSNÉ NÉMETH^{1,2}, László ZSAKÓ³

¹*Batthyány High School, Nagykanizsa, Hungary*

²*Doctoral School, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary*

³*Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary*

e-mail: erdosne@blg.hu, zsako@caesar.elte.hu

Abstract. Whether you teach programming or you are a competitive programmer yourself, one of the main questions is how to check the correctness of solutions. There are many different types of automatic judge systems both online and offline; you have to choose the appropriate one for the situation. In this paper, we discuss the types of judges and the possible feedbacks given by them. We also give an overview of the methods used by some well-known online sites from a pedagogical point of view. The technical issues of different grading systems are omitted.

Keywords: teaching programming, grading on algorithmic contests, automatic grading systems, ACM ICPC, IOI.

1. Overview

There are many papers about the online judges from a technical viewpoint: how these systems can be set up, what the problems with measuring the running times are, how to make good test files, etc. (Skupas, 2010; Maggiolo, 2012; Mares, 2009). There are papers from the point of view of a jury: is the solution correct, is it readable or not, (Skupiene, 2010), how can the solution's quality be measured, what are the differences between an accepted submission and a provably correct solution, what are the requirements of a correct task description. There are questions partly answered about manual grading in informatics contests (Pohl, 2008). There are papers about specific grading systems, like USACO and UVa. (Kolstad, 2007; Revilla *et al.*, 2008)

We think, all participants of the teaching and competing process (teachers, jury, contestants, students, companies) agree in the positive role of scientific, and especially informatics contests. According to Manev *et al.* (2009), the main reasons:

- It is more attractive for talented pupils when education lines up well with and is for the purposes of the participating in competitions.
- It is necessary to teach students to compete as early as possible.

In many countries there is a large gap between the knowledge and skills developed in the regular curriculum and the ones needed for algorithmic contests. So teaching

programming and also preparing for algorithmic contests is mostly an out-of-school activity. It needs more independent work from students and needs more specific attitudes from teachers.

A didactically and methodologically interesting question is how to check the solutions submitted by students to a given problem, while teaching programming to them; how to enable them to check their own solutions while practicing; and how to grade solutions in algorithmic contests. All three of these situations require different approaches.

2. Grading Systems Overview

There are many online and offline grading systems for evaluating the correctness of programs written by competitive programmers. These systems are very similar in the sense that they all use a set of test data files uploaded by task-setters, performing black-box testing. The programmers' code is tested automatically, by a code-checker – not by a human being – and contestants have to write their code accordingly. For each problem, there is a set of one or more input files and a set of corresponding correct output files or a specific judge tool which checks the output of the uploaded program. Each input file is according to strict specifications described in the problem statement. The program is run on each of those input files. In most cases, the output generated by the student's program must match the correct output exactly, to be accepted. The solution, in order to be evaluated as correct, needs to produce correct output, and run within specified time and memory constraints.

The test data should check

- The complexity of the algorithm used.
- The behaviour of the code on border cases.
- The memory consumption.
- The running time efficiency (constant) of the implementation.

The difference between the systems is the method of ranking, the grouping of test data files, and the feedback given.

In this paper, we discuss the different methods of automatic judging, the differences in the feedback, the question of using individual or grouped test cases, and the methods adopted by well-known online practice sites and online competitions. We concentrate on algorithmic tasks only, and we do not discuss the specialities of interactive and output only tasks. We compare different methods from a pedagogical viewpoint and concentrate on didactical aspects: which systems can be used for different age groups and in different phases of learning, practicing and competing.

3. Types of Grading

There are four different types of grading models, each named after the best-known contest or situation it is used in.

3.1. ACM-Style Grading

On ACM ICPC contests, only the perfect and optimally efficient solution is worth any points. When uploading a solution, two cases are possible:

- Accepted: your program is perfect, it runs within the time limit, doesn't exceed the memory limit and for every input it prints a correct output.
- Not accepted: something went wrong, the judging system gives back the first error code.

Usually the correct solution is worth one point.

In many ACM ICPC tournaments, the number of teams reaching a given score decreases exponentially in terms of the score.

Possibilities for differentiation between teams with the same number of correct solutions:

- Defining the difficulty of each task: X_i points for the correct solution of task i (instead of one point for all tasks).
- Assigning a solution time to every task: measuring the elapsed time for every task solved by a team or person. The solution times are summed to give a time penalty. In this case, competitors must recognize which tasks they can solve quickly (easy tasks) and which ones they can solve slowly (difficult tasks). The optimal strategy is to solve easy tasks first, and leave more time-consuming tasks to the end. For example, in the case of a 10-minute and a 50-minute task, starting with the easier one, the solution time is 10+60 minutes, while starting with the more difficult one, the solution time is 50+60 minutes.
- Sometimes there is a penalty for wrong submissions too: e.g. +20 minutes or $-X_i$ points for every incorrect solution submitted. In this case, the contestants are advised to test their solution offline at first.

This type of grading is acceptable only for older and well-practiced contestants (especially university students) who are:

- Strong enough to endure failure.
- Have enough routine to see what went wrong.
- Well-trained enough to be able to repair and to rethink their solutions again and again.

Apart from the ACM ICPC contest, the same grading is used in CodeChef Cook-off contests, CodeForces rounds and CSAcademy (with full feedback).

3.2. IOI-Style Grading

The test files are grouped at IOI. Test cases awaiting a solution with the same asymptotic complexity (e.g. $O(n)$, $O(n^*\log n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$) or logical complexity (e.g. problem restricted to an easier subproblem) are grouped together and worth a predetermined amount of partial points. The judge gives the partial points only if all the tests in the

group are passed, so only if the solution with the expected efficiency is perfect (or the solution solves the subproblem covered by the group of test cases).

The problem with the idea is that it contradicts the modern pedagogical principle of appreciating all the performance, rewarding all positive achievements. A fairness problem may arise as to whether a perfect $O(2^n)$ solution is better than an $O(n)$ solution that does not test a single special case and hence receives 0 points. But it is acceptable for such an international competition where the competitors are well-prepared and have advanced knowledge.

Apart from IOI, CodeChef Lunchtime uses this method for contests too.

3.3. National Olympiads

The modern pedagogical principle says that all performance and all success should be rewarded. The individually evaluated separate test cases give this sense of achievement to all contestants.

There must be targeted test cases prepared for:

- Default values described in a task description.
- Extreme values of the problem domain and range.
- Different methods.
- Typical errors.
- Large quantities of data.

The number of points achievable by different solutions – according to the abovementioned metrics – has to be decided in advance. The distribution of test cases has to be designed in such a way, that a good balance of different types of test cases is kept, and hence the judge assigns the desired scores to different partial solutions.

When practicing, this is the best method if the online judge gives back a very detailed, exact feedback for all test cases.

The national competitions, partly university and high school exams, the practice sites of online judges – including USACO, COCI, UVa and HackerEarth – also use this method.

3.4. Offline Judging

When somebody begins to learn programming, it is important to develop the right coding style and formatting; and all the small successes should be evaluated and appreciated. Beginners need clear, accurate, detailed (usually verbal) feedback about their work.

Checking the code on a computer is a wide-ranging activity for all competitors. Writing code on paper is crucial in sitting exams without computers.

When somebody wants to hack another person's solution, or wants to learn new methods, they must be able to read and understand other people's code.

When there is a penalty for wrong uploads, the importance of offline judging is crucial – everyone has to learn to read code and to create test data. Everybody has to test their solution for one or two given sets of test data and must learn to make their own test cases for different approaches.

All judges perform black-box testing only, but we think that in teaching programming and preparing for contests, white-box testing must also be used.

4. Test Data

The quality of the task description and the test cases is crucial for the quality of a contest or an online practice site. The evaluation summary presented in online judge systems should be easy to read and to understand.

Some problems can be easily solved using heuristic algorithms, some can be solved on a subset of test cases by generally incorrect algorithms, still receiving very good evaluation scores (Forišek, 2006), so it is very important, that test cases used by the judge have been carefully designed and checked as widely as they can be. Wide testing is limited by available resources and confidentiality (for live contests).

The speciality of CodeForces Educational Rounds is that the results that are obtained by the end of a round are preliminary, and after contest, there is a twenty-four-hour-period of open hacks, when every visitor may try to hack any complete solution to a problem from the round, and all successful hacks are added to the official test set and every solution is re-tested on the now extended test set. This method makes the test cases better after the contest, and it widens the group of possible test makers. This is good practice and it may be worthwhile to adapt it for other contests.

5. Feedbacks

In all automatic judging system after writing a program on the contestant's machine, the source code is uploaded to the judging server. The source code is compiled and run on the server. The automatic judge tests it with some inputs and outputs (string comparison) or with a specific judge tool (for more complicated problems). After the process, the server gives back simple or detailed feedback.

The simple feedback is: "accepted" or "not accepted" (in that case it gave back the first error message).

The detailed feedback given back is one the following. We show the notation used by almost all sites (HackerEarth, SPOJ, TopCoder, CSAcademy...), the one used by USA-CO, and the one used by CodeChef, respectively, in brackets for each possible feedback. Sometimes feedback is given for all the test cases, sometimes just till the first error.

- Accepted (AC, *, ): the program is correct. It produced the right answer within the time and memory bounds.

- Wrong Answer (WA, x,): correct solution not reached for the inputs. This may include empty or missing output as well.
- Compile Error (CE, c,): the compiler could not compile the uploaded program. Warning messages are ignored. Usually the compiler output messages are reported on the screen.
- Runtime Error (RE, !,): the uploaded program failed during execution (segmentation fault, floating point exception...). Sometimes the exact cause is not reported to the user, sometimes it is specified: SIGSEGV: segmentation fault, SIGABRT: fatal error, SIGXFSZ: output is too large, NZEC: non-zero exit code, SIGFPE: floating point error.
- Time Limit Exceeded (TL-TLE, t,): the program did not terminate within the time limit. This error does not give information about whether the program would have reached the correct solution or not.
- Memory Limit Exceeded (ML-MLE, !, -): the uploaded program tried to use more memory than the judge allows. This is hard to separate from Runtime Errors for technical reasons, hence some judges do not report this.

Some systems also give other verdicts (UVa):

- Output Limit Exceeded (OL-OLE): The program tried to write too much information. This usually occurs if it goes into an infinite loop.
- Submission Error (SE): The submission is not successful. This is due to some error during the submission process, or data corruption.
- Presentation Error (PE): The program outputs are correct, but outputs are not presented in the correct way. However, usually the judging systems ignore extra white spaces, like '\n', '\t'.
- Restricted Function (RF): The uploaded program is trying to use a function which may be harmful to the system.
- In Queue (QU): The judge is busy and cannot attend the submission. It will be judged as soon as possible.
- Cannot Be Judged (CJ): The judge doesn't have test input and outputs for the selected problem.

6. Conclusions

Our opinion is that partial scoring and detailed feedback is a must at the beginning of the learning process of programming. White-box testing with verbal feedback about the coding style is an optimal case for future work.

While practicing, detailed feedback is also a must. But instead of a simple “Wrong Answer” feedback, the detailed error signals are very useful. For example, if the task is about finding the shortest way in a graph, the type of error would be textually in the feedback, like:

- Wrong path length.
- The given path does not have minimal length.

- An invalid vertex is on the given path.
- One vertex turns up several times on the given path.
- The end of the given path is not the end of the expected path.
- The given sequence of vertices is not a path in the graph.

If the error is noticeable, then the judge should give the solution back with the cause of the error.

Our opinion – supported by our practice with high school and university students too – is, that feedback of tests case by case is a must on exams and on national Olympiads level as well.

When students are well-practiced on international level – like IOI and ACM – is the first level at which test cases in groups are acceptable. These students' knowledge can be measured well through the carefully selected tasks with adequate points worth, well grouped test cases.

From pedagogical viewpoint the ACM-style evaluation, differentiating based on elapsed time is good for easy, have-to-solve tasks only, not for making contests for high school students.

References

- CodeChef, not-for-profit educational initiative by Directi. <https://www.codechef.com>
- CodeForces online task archive and contest site. <http://codeforces.com>
- COCI Croatian Open Competition in Informatics. <http://hsin.hr/coci/>
- Cormack, G. (2006). Random factors in IOI 2005 test case scoring. *Informatics in Education*, 5(1), 5–14.
- CSAcademy educational platform. <https://csacademy.com/>
- Forišek, M. (2006). On the suitability of programming tasks for automated evaluation. *Informatics in Education*, 5(1), 63–75.
- HackerEarth: Be a better programmer! <https://www.hackerearth.com/>
- Horváth, Gy. (2014). A programozási versenyek szerepe az oktatásban. *INFOÉRA Konferencia 2014*.
- Kolstad, R., Piele, D. (2007). USA Computing Olympiad (USACO). *Olympiads in Informatics*, 1, 105–111.
- Manev, K., Sredkov, M., Bogdanov, T. (2009). Grading systems for competitions in programming. *Proceedings of the XXXVIII. Spring Conference of the Union of Bulgarian Mathematicians*, 2009.
- Mares, M. (2009). Moe—Design of a Modular Grading System. *Olympiads in Informatics*, 3, 60–66.
- MESTER online task archive and judge system. <https://mester.inf.elte.hu>
- Maggioli, S., Mascellani, G. (2012). Introducing CMS: A Contest Management System. *Olympiads in Informatics*, 6, 86–99.
- Pohl, W. (2008). Manual Grading in an Informatics Contest. *Olympiads in Informatics*, 2, 122–130.
- Revilla M.A., Manzoor, S., Liu R. (2008). Competitive learning in informatics: the UVa on-line judge experience. *Olympiads in Informatics*, 2, 131–148.
- Skupas, B. (2010). Feedback Improvement in Automatic Program Evaluation Systems. *Informatics in Education*, 9(2), 229–237.
- Skupiene, J. (2010). Improving the Evaluation Model for the Lithuanian Informatics Olympiads. *Informatics in Education*, 9(1), 141–158.
- SPOJ Sphere Online Judge. <http://www.spoj.com/>
- TopCoder Algorithms&Analytics. <https://www.topcoder.com>
- USACO USA Computing Olympiad open contest and training pages. <http://usaco.org/>
- UVa Online Judge. <https://uva.onlinejudge.org>
- Verhoeff T.(2008). Programming task Packages: Peak exchange Format, *Olympiads in Informatics*, 2, 192–207.



Á. Erdősné Németh teaches mathematics and informatics at Batthyány Lajos High School in Nagykanizsa, Hungary. A lot of her students are in the final rounds of the national programming competitions, some on CEOI and IOI. She is a PhD student in the Doctoral School of Faculty of Informatics, *Eötvös Loránd* University in Hungary. Her current research interest is teaching computer science for talented pupils in primary and secondary school.



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

The Next Course of Study from 2022 and a History of the Subject “Informatics” in Japanese High Schools

Yoshiaki NAKANO¹, Katsunobu IZUTSU²

¹*Kobe Municipal High School of Science and Technology, Japan*

²*Hokkaido University of Education, Japan*

e-mail: info@nakano.ac, idutsu@gmail.com

Abstract. In Japan, all senior high schools have had the subject “Informatics” since 2003. Before that, there were not any regular classes for informatics in Japanese senior high schools. Japanese Course of Study is revised approximately every ten years. The current one was implemented in 2013; the next one, published on March 2018, will be implemented in 2022. The authors outline the history as well as the prospect from 2022 on of the information studies education in Japanese senior high schools.

Keywords: general subject “Informatics”, course of study, scientific understanding of information.

1. The Japanese Course of Study

The Japanese Course of Study is published by Japanese Ministry of Education. With all the controversies of its legal binding power, it has tremendous influence on school education.

In Japan, all senior high schools have had the general subject “Informatics” since 2003, before which there were not any classes of information studies as the general education in Japanese senior high schools. Japanese Course of Study has been revised approximately every ten years and the current one has been implemented since 2013. The next Course of Study, which will be implemented from 2022, was published on March 2018.

2. Three Objectives of Information Studies Education in Primary and Secondary Education

The information studies education in Japanese primary and secondary education encourages pupils and students to develop their “Practical skills in using information actively”, “Scientific understanding of information” and “Positive attitudes towards today’s information-laden society”. The three objectives were defined in the first report of “the Researchers Conference for Promotion of K-12 Information Studies Education”, published in October 1997. Based on the report, the previous Course of Study launched its version of information studies education. (MEXT, 1997)

The three objectives are defined as follows:

- “Practical skills in using information actively”: The ability to use information means to collect, evaluate, express, process, and/or create the information needed, and to dispatch and communicate it in consideration of the receiver.
- “Scientific understanding of information”: Understanding the characteristics of information means to use information actively and to understand basic theories and methods for handling information appropriately and for evaluating and improving one’s own use of information.
- “Positive attitudes towards today’s information-laden society”: The positive attitude that should be accompanied by the intention to understand the role and influence of information in our life, to consider the importance of information ethics and one’s own responsibility for information, and to participate actively in creating a desirable society.

3. Information Studies Education in Elementary Schools

Elementary schools do not provide any specific subject of information studies but encourages pupils to develop their skills in using information actively by means of information devices like personal computers in class activities centered around the subject “Integrated Studies” (Nakano and Izutsu, 2013).

The next Course of Study for Elementary Schools describes what needs to be taken into account when making teaching plans in its Chapter 1, the general provisions (MEXT, 2017a):

In order to foster the ability to utilize information, each school should prepare the necessary environment to utilize information devices such as computer and networks, and enrich learning activities appropriately utilizing them.

To implement the next learning activity systematically.

- 1) Basic operations of computer such as keyboard operation.
- 2) Logical thinking skills by experience of computer programming.

What makes a great difference from the current Course of Study is “experience of computer programming”.

Moreover, the next Course of Study states on “Integrated Studies”:

It comprises the process of explorative learning, in which pupils participate in activities such as collecting information, processing and sending out information, utilizing computers and networks properly and effectively. In doing so, pupils learn about the basic operation of the computer to obtain the information needed and how to make a subjective or spontaneous choice among different information means.

4. Information Studies Education in Junior High Schools

In junior high schools, the subject of Technology and Home Economics generally covers information studies. The next Course of Study classifies the field of Technology as follows (MEXT, 2017b):

- A. Technology of materials and their processing.
- B. Technology of animal and plant growth.
- C. Technology of energy conversion.
- D. Technology of information.

The content of D “Technology of information” is defined as:

- (1) Information technology to support our life and society.
- (2) Interactive content by computer programs.
- (3) Measurement and control by computer programs.
- (4) Development of society with information technology.

Half of the 175 hours of the subject Technology and Home Economics are allotted to the content of Technology. The learning of information used to take up half the content, but now takes up only a quarter of it. Thus the time period for information studies that formerly reached 44 hours currently amounts to only 22 hours. This means a significant change in quantity as well as quality.

5. Information Studies Education in Senior High Schools

In senior high schools, the subject “Informatics” chiefly serves for information studies education. It is comprised of General Informatics (Informatics as general subject) and Major Informatics (Informatics as major subject).

5.1. The Subject General Informatics

Introduction of the Subject General Informatics and Previous Course of Study

The subject “Informatics” was newly established in high schools in 2003. General Informatics and Major Informatics were set up, and General Informatics are composed by “Information A”, “Information B” and “Information C”. “Information A” teaches the practical skills in using information actively, “Information B” deals with the sci-

tific understanding of information, and “Information C” nurtures the positive attitudes towards today’s information-laden society. Students are supposed to select one of these three. It would have been ideal that students themselves could freely select the one suitable to their characteristics and their future career. However, the actual situation of most schools was that each school designated the subject with no regard to the students’ preference. The rate of “Information A” was 80% of high schools nationwide. On the other hand, “Information B” was only 5% and “Information C” was only 15%. Unfortunately, most of those subjects seems to have been subject to the preconception of “the practical skills in using information actively = the skills of using office software” (MEXT, 1999). The classes accordingly limited themselves to exercises in using word processors and presentation software.

Information A:

- (1) Devices for active use of information and information equipment.
- (2) Collection and transmission of information and utilization of information equipment.
- (3) Integrated processing of information and utilization of computers.
- (4) Development of information equipment and changes in our lives.

Information B:

- (1) Solving Problems and utilizing Computers.
- (2) Structure and function of computer.
- (3) Modeling problems and solving by computers.
- (4) Information technology supporting the information-laden society.

Information C:

- (1) Digitization of information.
- (2) The Internet and communication.
- (3) Collection and transmission of information and personal responsibility.
- (4) Progress of information technology and its impact on society.

The problems with information studies education in those days were primarily attributable to the inadequacy in the teacher training of the subjects. As such training, “Teachers’ license program of the new subject Informatics for in-service teachers” was implemented over the three years since 2000, and the teachers who completed a fifteen-day course acquired its certificate and were in charge of the pertinent subjects and classes. The authors have to say that it was extremely difficult for teachers to acquire the knowledge and skills necessary for teaching the appropriate contents of Informatics in no more than fifteen days.

Present Situation of Informatics Teaching and Current Course of Study

In spite of those problems, it was meaningful that Informatics was established as a new subject. It encouraged a lot of research groups of Informatics to be created in each prefecture, and they were eventually put together into **Japan Informatics Teachers' Association (JITA)**. Every year sees in its annual conference not only high school teachers but also university professors holding research presentations. Those active comprises a driving force for improving the lesson and creating new initiatives of Informatics.

The current Course of Study, being applied to the students who entered high schools in 2013 and later, reorganized General Informatics into two subjects: "Society and Information" and "Information Science". Their contents are as follows (MEXT, 2009):

Society and Information:

- (1) Active use of information and its expression.
- (2) The Internet and communication.
- (3) Issues of today's information-laden society and information ethics.
- (4) Construction of desirable information societies.

Science of Information:

- (1) Personal computers and the Internet.
- (2) Problem solving and effective use of personal computers.
- (3) Information management and problem solving.
- (4) Progress of information technology and information ethics.

The subject corresponding to "Information A" disappeared, the "Society and Information" was substituted for "Information C", and "Information B" developed into the "Information Science". Nevertheless, these reformations did not change the situation in which each school designated which subject their students should take; the 80% of high schools assigned "Society and Information", whereas only 20% designated "Information Science".

Moreover, even though more than ten years have passed since Informatics was newly established, the situation of teaching Informatics is far from having been improved; rather it is in a much poorer condition. Of Informatics teachers nationwide, only 20% is dedicated, 50% teaches other subjects concurrently, and the remaining 30% do not have an Informatics teacher's license, only with a temporary license.

Future of Informatics and Next Course of Study

Third stage of revision of the Course of Study changes General Informatics dynamically rather than remaining as a minor improvement of it. The previous Course of Study and the current one both presupposed a choice among the required subjects: "Information A", "Information B" and "Information C", or "Society and Information" and "Information Science" and did not provide any advanced subject in General Informatics. However, in the next Course of Study, the compulsory subject is unified into "Informatics I" and, furthermore, "Informatics II" has been set as an advanced subject (Fig. 1) (MEXT, 2018).

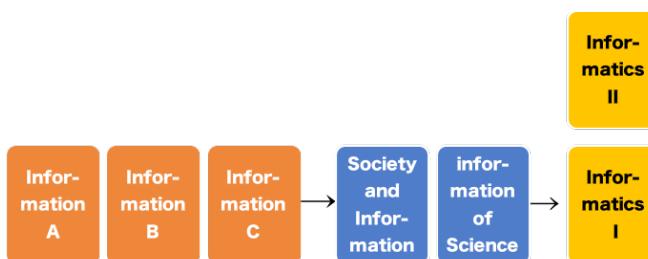


Fig. 1. Historical shift in the subjects and levels of General Informatics.

Aim of General Informatics

“Informatics I” and “Informatics II” provide high school students with learning activities in which they can make use of scientific viewpoints and ways of thinking about information to discover and solve various problems by utilizing information technology appropriately and effectively. The subject aims to nurture the propensity and ability to participate actively in society as follows:

- (1) To understand more about information and information technology and methods to discover and solve various problems by using them, to acquire the skills needed and to deepen their understanding of the relationship between the information society and people therein.
- (2) To understand various events in terms of information and its connection to them, cultivate the ability to appropriately and effectively utilize information and information technology toward finding and solving problems with the events.
- (3) To appropriately utilize information and information technology and cultivate an attitude to participate actively in the information society.

Contents of “Informatics I” and “Informatics II”

The contents of the relevant two subjects are as follows:

Informatics I:

- (1) Problem solving of information society:
characteristics of information and media, utilization of information technology, problem discovery, problem solving, legal system, information security, personal responsibility, information ethics, and impact on society
- (2) Communication and information design:
characteristics of media, communication means, effective communication, and information design
- (3) Computer and programming:
computer mechanism, algorithm, modeling, simulation, evaluation, and programming
- (4) Utilization of information communication network and data:
Internet structure, protocol, database, and information system

Informatics II:

- (1) Progress of information society and information technology:
development of information technology, future information technology, and information society
- (2) Communication and contents:
type of communication, characteristics of media, multimedia contents production, and information design
- (3) Information and data science:
data science, database, modeling, and data processing
- (4) Information system and programming:
information system, information service, information security, software development process, and project management

(5) Exploration of problem solving using information and information technology:

imagination of new values, and effective utilization of information technology

As we noted above, "Informatics II" is positioned above "Informatics I". The correspondence between the contents of the two subjects ("Informatics I" → "Informatics II") is fairly clear: (1) → (1), (2) → (2), (3) → (4), (4) → (3). (Fig. 2) (5) of "Informatics II" amounts to exploring comprehensive tasks utilizing the knowledge and skills acquired in "Informatics I" and "Informatics II". We can see it as corresponding to project research in Major Informatics, which we will see below.

Although the standard unit number of each subject remains two, the flat organization of Informatics teaching has now grown into a "two-story structure" with one based on the other. The compulsory subjects one of which students were required to choose are unified in Informatics I, and Informatics II opens the door to further and deeper leaning. This structural change is likely to improve the presence of Informatics at high school.

A slightly more concrete look at the contents will show that information design, programming, networking, information security, and databases are dealt with in "Informatics I". In addition, "Informatics II" enriches the contents related to data science and information systems. Placing emphasis on the scientific understanding of information, it adopts further contents being dealt with in the current Major Informatics.

There may arise a concern about the teachers, whose specialties are other subjects or who don't have an Informatics teacher's license: can they appropriately teach these contents? It can be a relieving fact that more and more university professors are actively incorporating practical training, while an increasing number of Boards of Education are employing Informatics teachers. Additional in-service training is also supporting Informatics teachers and improving their teaching environment. Educational equipment such as computers and networks must also be improved for their active and effective practices.

Informatics I		Informatics II
(1) Problem solving of information society	→	(1) Progress of information society and information technology
(2) Communication and information design	→	(2) Communication and contents
(3) Computer and programming	→	(3) Information and data science
(4) Utilization of information communication network and data	→	(4) Information system and programming
		(5) Exploration of problem solving using information and information technology

Fig. 2. Content correspondence between Informatics I and II.

5.2. *The Subject Major Informatics*

Current situation

Alongside of "General Informatics", the current Course of Study has provided 13 subjects of Major Informatics with senior high schools.

However, only a very small number of schools teach subjects of Major Informatics. Japan does not have more than 20 senior high schools that offer the course Major Informatics.

Aim

Major Informatics provides high school students with the practical and experiential learning activities necessary for the acquisition of scientific viewpoints and ways of thinking on information. This is because students will need these to realize a healthy and sustainable development of local society and information society, including the information industry. The subject aims to help students to acquire abilities as follows:

- (1) To understand each field of information systematically, and to obtain related skills.
- (2) To discover issues relating to the information industry and cultivate the ability to solve reasonably and creatively based on the ethical sense required of workers.
- (3) To nurture the rich human nature necessary for professionals, learn by themselves to build a better society, and cultivate an attitude to actively and cooperatively work on the creation and development of the information industry.

The difference in objectives between General Informatics and Major Informatics is that this latter focuses on "practical and experiential learning activities", "a sound and sustainable development of information society by the information industry", and "professional ethics".

New Subjects of Major Informatics and their Characteristics

The new Course of Study reorganizes subjects of Major Informatics by introducing some new subjects as well as revising the overall content of the subjects. This reorganization is expected to help foster qualified young people who have the creative capability of solving various problems in addition to understanding occupational ethics.

The new Course of Study divides Major Informatics into 12 different subjects in reference to four characteristics: basic, system design and management, creation and production of information content, and synthetic.

- Basic subjects:
"Information Industry and Society", "Expression and Management of Information", "Information Technology", and "Information Security"
- System design and management subjects:
"Programming for Information Systems", "Network System", and "Database"
- Creation and production of information content subjects:
"Information Design", "Production and Dissemination of Contents", and "Media and Services"

- Synthetic subject:
“Informatics Practice” and “Project Research”

5.3. Other Subjects Related to Informatics in Special High Schools

The Course of Study notified in 1979 provided some major subjects of industry and commerce, which by and large corresponded to the present-day information studies. Subsequently, the ministry of education established the course of information technology in high schools of industry and that of **information processing in high schools of commerce**, which have played a central role of information studies education in the nation.

Other special high schools teach some other subjects of information: “Agriculture and Information” in agricultural schools, “Maritime Information Technology” in fishery schools, “Nursing Information” in nursing schools, and “Welfare Information” in welfare schools.

Likewise, the new Course of Study provides “Industrial Information Mathematics”, “Programming Technology”, “Hardware Technology”, “Software Technology” and “Computer Systems Technology” for high schools of industry and “Information Processing”, “Software Application”, “Programming”, “Network Application” and “Network Management” for high schools of commerce. The subjects will be learned by a lot of students in a wide range of high schools.

6. Trends in General Informatics

The characteristics of General Informatics in the three sets of Course of Study can be summarized with respect to the three objectives as in Table 2.

The third revision of Course of Study will hopefully allow us finally to start ideal information studies education in Japanese senior high schools. However, further monitoring and support will be necessary to see if it will really be realized. The authors would also like to participate continuously in the practice, observation, and support of this new endeavor of information studies education in Japan.

Table 2
Specific gravity of the three objectives in General Informatics

	Previous	Current	Next
Practical skills in actively using information	****	**	*
Positive attitude toward participation in today's information-laden society	**	***	**
Scientific understanding of information	*	*	****

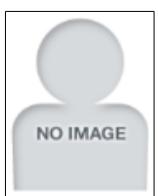
The number of asterisks in the table indicates that the objective is pursued actively (****), moderately (**), or barely (*).

References

- MEXT* (1997). *The Researchers Conference for Promotion of K-12 Information Studies Education: For implementation of systematic information studies education* (The first report).
http://www.mext.go.jp/b_menu/shingi/chousa/shotou/002/toushin/971001.htm
- MEXT (1999). *Course of Study for senior high schools.*
http://www.mext.go.jp/a_menu/shotou/cs/1320221.htm
- MEXT (2009). *Course of Study for senior high schools.*
http://www.mext.go.jp/a_menu/shotou/new-cs/youryou/1304427.htm
- MEXT (2017a). *Course of Study for elementary schools.*
http://www.mext.go.jp/component/a_menu/education/micro_detail/_icsFiles/afieldfile/2018/05/07/1384661_4_3_2.pdf
- MEXT (2017b). *Course of Study for junior high schools.*
http://www.mext.go.jp/component/a_menu/education/micro_detail/_icsFiles/afieldfile/2018/05/07/1384661_5_4.pdf
- MEXT (2018). *Course of Study for senior high schools.*
http://www.mext.go.jp/component/a_menu/education/micro_detail/_icsFiles/afieldfile/2018/04/24/1384661_6_1.pdf
- Nakano, Y., Izutsu, K. (2013). The new Course of Study and a prospect of information studies education in Japan. In: I. Diethelm *et al.* (Eds.) *Informatics in Schools: Local Proceedings of the 6th International Conference ISSEP 2013; Selected Papers; Oldenburg, Germany, February 26–March 2, 2013*. Universitätsverlag Potsdam, 89–96.



Y. Nakano received his B.Eng., M.Eng. from Shibaura Institute of Technology in 1988, and 1990, respectively. He is the national certified Professional Engineer of Engineering Management and Information Engineering. He is a teacher in Kobe Municipal High School of Science and Technology. His research interests are on informatics education in secondary schools, teacher development and universities entrance exam.



K. Izutsu, M.A. and PhD (Hokkaido University), is Associate Professor of linguistics at Hokkaido University of Education. His research interests include cognitive and functional approaches to language, linguistic pragmatics, anthropological linguistics, information and communication technology, multimodal communication, language learning and acquisition, and human evolution and language development.

* MEXT – Ministry of Education, Culture, Sports, Science & Technology

Current Situation of Teachers of Informatics at High Schools in Japan

Yasuichi NAKAYAMA^{1*}, Yoshiaki NAKANO², Yasushi KUNO¹,
Ben Tsutom WADA³, Hiroyasu KAKUDA¹, Masami HAGIYA⁴,
Katsuhiko KAKEHI⁵

¹*The University of Electro-Communications
Chofu, Tokyo 182-8585 JAPAN*

²*Kobe Municipal High School of Science and Technology*

³*Nagano University*

⁴*The University of Tokyo*

⁵*Waseda University, currently with Tokyo Online University*

e-mail: nakayama@uec.ac.jp, info@nakano.ac, y-kuno@uec.ac.jp,
{wadaben, kakuda}@acm.org, hagiya@is.s.u-tokyo.ac.jp, kakehi@waseda.jp

Abstract. In March 2018, the Japanese Ministry of Education, Culture, Sports, Science and Technology revised the curriculum guidelines for high school, which will be applied in 2022. The subject of Informatics has been drastically changed; ‘Informatics I’ and ‘Informatics II’ have a predominantly scientific approach. This could be problematic given that a lot of ‘temporary teachers’ and ‘teachers without a proper license’ teach Informatics, and more than half of the teachers that teach Informatics are in charge of multiple subjects. So, it may be difficult to implement new curriculum in the dozens of prefectures that have few teachers who specialize in Informatics. We report on the problems our investigation revealed.

Keywords: informatics education, elementary and secondary education, teacher license.

1. Introduction

In 2003, the subject of Informatics was introduced in Japanese high schools. The Ministry of Education, Culture, Sports, Science and Technology (MEXT) revised curriculum guidelines in 2013. Under the current curriculum guidelines, each high school adopts either ‘Information study for participating community’ or ‘Information study by scientific approach’.

In March 2018, MEXT revised the curriculum guidelines for high schools further (MEXT, 2018), (Kano, 2017). The latest curriculum guidelines will be applied in 2022. The subject of Informatics has been drastically changed, with ‘Informatics I’ and ‘Informatics II’ having a mainly scientific approach.

* Corresponding Author

Informatics is a field of science that investigates principles and technologies for defining semantics, creating value, and giving order to the world, by processing information. The principles and technologies for processing information include those of creation, generation, collection, representation, recording, recognition, analysis, transformation, and transmission (Hagiya, 2015). In order to teach Informatics to students at high schools, a wide range of knowledge and skills related to Informatics are required. Licensed teachers who specialize in the subjects related to Informatics are necessary.

However, there are many unlicensed teachers who teach Informatics. Researchers in educational administration have brought attention to this since Informatics first started being taught at high schools, but there has been little improvement (Nakano, 2006), (Kano, 2012), (Hagiya, 2016).

In 2003 when subject of Informatics was introduced, no licensed teachers existed because the subject did not exist until then. Since then, special rules have been applied to teachers assigned to teach Informatics. One of the special rules is a training course held from 2000 to 2003. A teacher who holds a license for another subject such as Mathematics, Science, etc. can obtain a license for Informatics by training for 15 days. Approximately 14,000 teachers have acquired licenses for Informatics.

Other special rules are ‘temporary teachers’ and ‘teachers without a proper license’. These rules can be applied to other subjects, but since 2003 they have been widely used for Informatics. A prefectural board of education can hire ‘temporary teachers’ for up to three years if they cannot employ teachers with a proper license. In addition, the prefectural board of education can allow ‘teachers without a proper license’ to teachers who have a license for another subject.

In 2015, we requested the disclosure of administrative documents from the MEXT in order to obtain the number of ‘temporary teachers’ and ‘teachers without a proper license’ in 47 prefectures. So far, we have learned that a lot of ‘temporary teachers’ and ‘teachers without a proper license’ are applied for Informatics in comparison with other subjects (Nakayama *et al.*, 2017).

In this paper, we report the results of our latest request for disclosure of administrative documents from 47 prefectures and 19 major cities. It became clear that there are only a few teachers who specialize in Informatics (i.e., teachers who only teach Informatics) even among those who hold proper licenses for it, and more than half of the teachers who teach Informatics cover multiple subjects.

2. Procedure for Obtaining Disclosure of Administrative Documents and the Results

We have collected data on whether holders of Informatics licenses teach only Informatics or whether they also teach other subjects than Informatics.

In June 2015, MEXT required prefectures and major cities to report the recruitment, assignment, and training of teachers for Informatics. We collected the reports of 47 prefectures and 19 major cities. Although we have 20 major cities in Japan, we did not request disclosure for Sagamihara City, as they have no municipal high school.

Electronic applications were available in 44 prefectures and 18 cities. Data was provided electronically from 16 prefectures and 1 city, and paper documents were provided from the remaining prefectures and cities.

In showing how many teachers taught Informatics, only teachers at public schools were counted. National schools, private schools, schools with correspondence courses, and special support schools were excluded. Table 1 and Table 2 show the number of teachers who teach Informatics in 47 prefectures and 19 major cities as of May 2015.

Table 1
Teachers assigned for Informatics (Prefectures, May 2015)

Prefecture	Full-time teachers who teach Informatics					Public schools	
	Total	Teachers with proper license		Temporary teachers	Teachers without proper license		
		Assigned only for Informatics	In charge of multiple subjects				
1 Hokkaido	289	37	134	0	118	237	
2 Aomori	59	17	22	11	9	50	
3 Iwate	99	2	47	15	35	74	
4 Miyagi	101	13	43	3	42	76	
5 Akita	75	12	42	13	8	58	
6 Yamagata	45	3	39	0	3	28	
7 Fukushima	120	9	50	61	0	96	
8 Ibaraki	114	5	43	15	51	79	
9 Tochigi	132	4	20	108	0	52	
10 Gunma	130	3	111	3	13	63	
11 Saitama	231	210	21	0	0	131	
12 Chiba	327	22	305	0	0	124	
13 Tokyo	163	163	0	0	0	233	
14 Kanagawa	316	66	203	0	47	137	
15 Niigata	126	0	58	3	65	95	
16 Toyama	83	3	37	2	41	44	
17 Ishikawa	105	1	25	20	59	45	
18 Fukui	40	1	26	12	1	40	
19 Yamanashi	50	1	27	0	22	30	
20 Nagano	213	9	60	0	144	102	
21 Gifu	149	5	56	5	83	48	
22 Shizuoka	126	11	66	0	49	108	
23 Aichi	435	57	152	1	225	176	
24 Mie	82	30	40	0	12	66	
25 Shiga	174	4	76	5	89	42	
26 Kyoto	57	10	47	0	0	43	
27 Osaka	266	140	116	0	10	142	
28 Hyogo	294	74	105	0	115	116	
29 Nara	67	10	35	11	11	33	
30 Wakayama	86	7	30	11	38	46	
31 Tottori	16	4	11	0	1	19	
32 Shimane	53	2	39	0	12	38	

Continued on next page

Table 1 – continued from previous page

Prefecture	Full-time teachers who teach Informatics					Public schools	
	Total	Teachers with proper license		Temporary teachers	Teachers without proper license		
		Assigned only for Informatics	In charge of multiple subjects				
34 Hiroshima	84	7	50	1	26	99	
35 Yamaguchi	49	0	49	0	0	38	
36 Tokushima	61	0	46	1	14	31	
37 Kagawa	43	6	36	1	0	30	
38 Ehime	90	0	88	0	2	51	
39 Kochi	56	0	14	12	30	50	
40 Fukuoka	123	13	93	4	13	87	
41 Saga	45	9	30	0	6	22	
42 Nagasaki	61	4	32	8	17	51	
43 Kumamoto	48	17	20	7	4	45	
44 Oita	42	17	4	5	16	35	
45 Miyazaki	47	3	17	17	10	23	
46 Kagoshima	92	1	48	36	7	48	
47 Okinawa	48	39	1	0	8	45	

Table 2
Teachers assigned for Informatics (Major Cities, May 2015)

City	Full-time teachers who teach Informatics					Public schools	
	Total	Teachers with proper license		Temporary teachers	Teachers without proper license		
		Assigned only for Informatics	In charge of multiple subjects				
48 Sapporo	9	7	2	0	0	7	
49 Sendai	3	1	2	0	0	3	
50 Saitama	4	4	0	0	0	4	
51 Chiba	7	0	4	0	3	2	
52 Kawasaki	12	3	3	0	6	6	
53 Yokohama	28	25	1	0	2	9	
54 Sagamihara	0	0	0	0	0	0	
55 Niigata	7	0	4	3	0	3	
56 Shizuoka	3	0	3	0	0	3	
57 Hamamatsu	1	1	0	0	0	1	
58 Nagoya	12	9	3	0	0	15	
59 Kyoto	28	6	22	0	0	11	
60 Osaka	27	4	11	0	12	20	
61 Sakai	2	0	2	0	0	1	
62 Kobe	23	4	19	0	0	7	
63 Okayama	1	0	1	0	0	1	
64 Hiroshima	4	0	4	0	0	4	
65 Kitakyushu	1	0	1	0	0	1	
66 Fukuoka	5	0	5	0	0	3	
67 Kumamoto	1	0	0	1	0	2	

Table 3
Teachers assigned for Informatics (47 prefectures and 19 cities, May 2015)

	Full-time teachers who teach Informatics				Public schools	
	Total	Teachers with proper license		Tem- porary teachers		
	Assigned only for Informatics	In charge of multiple subjects				
47 prefectures	5588 (100)	1059 (19.0)	2657 (47.5)	395 (7.1)	1477 (26.4) 3368	
19 cities	178 (100)	64 (36.0)	87 (48.9)	4 (2.2)	23 (12.9) 103	
Total	5766 (100)	1123 (19.5)	2744 (47.6)	399 (6.9)	1500 (26.0) 3471	

Table 3 shows the total number for prefectures and major cities. We confirmed that a lot of ‘temporary teachers’ and ‘teachers without a proper license’ teach Informatics, as reported in (Nakayama *et al.*, 2017).

In 47 prefectures and 19 major cities, as shown in Table 3, about thirty percent of full-time teachers are ‘temporary teachers’ or ‘teachers without a proper license’. In addition, about fifty percent of full-time teachers are assigned to multiple subjects, and only about twenty percent of full-time teachers specialize in Informatics.

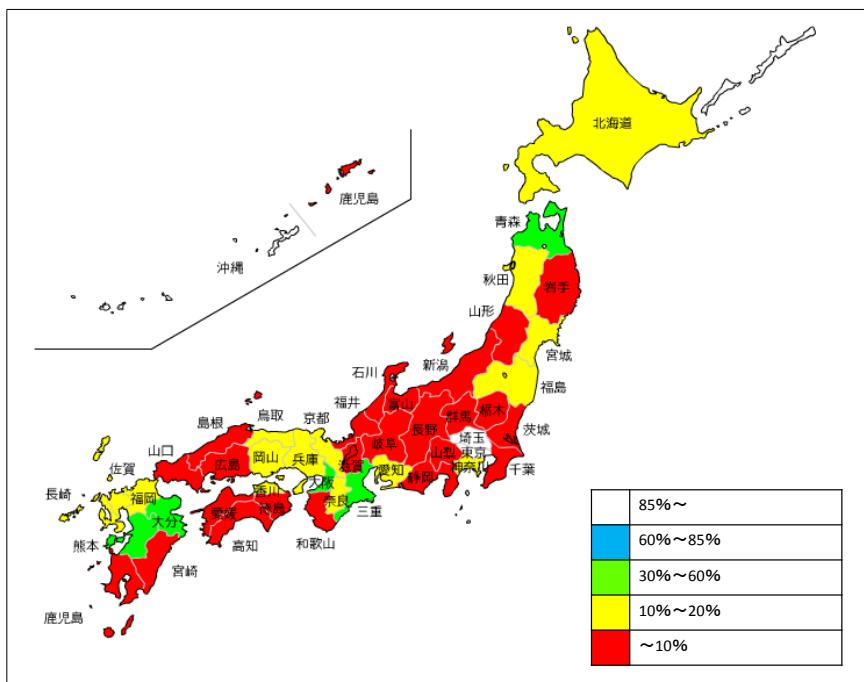


Fig. 1. Ratio of teachers who specialize in Informatics (May 2015).

The map in Fig. 1 shows the ratio of teachers who specialize in Informatics. The number of teachers who specialize in Informatics is more than half of the full-time teachers in Saitama Prefecture, Tokyo Prefecture, Osaka Prefecture, Okinawa Prefecture, Sapporo City, Saitama City, Yokohama City, Hamamatsu City, and Nagoya City (Table 4). The number of teachers who specialize in Informatics is more than half of the schools in Saitama Prefecture, Tokyo Prefecture, Osaka Prefecture, Hyogo Prefecture, Okinawa Prefecture, Sapporo City, Saitama City, Kawasaki City, Yokohama City, Hamamatsu City, Nagoya City, Kyoto City, and Kobe City (Table 5). These prefectures and cities are located in urban areas, except for Okinawa Prefecture.

The prefectures and cities that are not listed in Table 4 and Table 5 have zero or few teachers who specialize in Informatics. Many of these prefectures and cities are located in rural areas.

Table 4
Teachers who specialize in Informatics per full-time teachers (May 2015)

	Teachers who specialize in Informatics	Full-time teachers	Ratio
13 Tokyo Prefecture	163	163	1.00
50 Saitama City	4	4	1.00
57 Hamamatsu City	1	1	1.00
11 Saitama Prefecture	210	231	0.91
53 Yokohama City	25	28	0.89
47 Okinawa Prefecture	39	48	0.81
48 Sapporo City	7	9	0.78
58 Nagoya City	9	12	0.75
27 Osaka Prefecture	140	266	0.53

Table 5
Teachers who specialize in Informatics per schools (May 2015)

	Teachers who specialize in Informatics	Schools	Ratio
53 Yokohama City	25	9	2.78
11 Saitama Prefecture	210	131	1.60
48 Sapporo City	7	7	1.00
50 Saitama City	4	4	1.00
57 Hamamatsu City	1	1	1.00
27 Osaka Prefecture	140	142	0.99
47 Okinawa Prefecture	39	45	0.87
13 Tokyo Prefecture	163	233	0.70
28 Hyogo Prefecture	74	116	0.64
58 Nagoya City	9	15	0.60
62 Kobe City	4	7	0.57
59 Kyoto City	6	11	0.55
52 Kawasaki City	3	6	0.50

3. Discussion

As shown in Section 2, there are more Informatics teachers who specialize in Informatics in urban areas than in rural areas. In particular, a lot of ‘temporary teachers’ and ‘teachers without a proper license’ teach Informatics at high schools in rural areas.

In February 2018, MEXT announced the situation of ‘teachers without a proper license’ as of May 2017; across Japan 1,148 of 3,077 ‘teachers without a proper license’ were assigned for Informatics. Licensed subjects for the above-mentioned 1,148 teachers are shown in Fig. 2. We find that teachers of another subject, such as Art, Geography and History, Home economics, Foreign languages, Health and Physical education, etc., teach Informatics.

In addition, more than half of the teachers with a proper license for Informatics teach multiple subjects as shown in Section 2. Teachers in charge of multiple subjects at high schools may be unaware that they are Informatics teachers and are responsible for Informatics education, so there is a concern that they will not participate in study meetings for Informatics, such as *Zenkojoken* (the study group for high school Informatics teachers across Japan, <http://www.zenkojoken.jp>).

In 2013, the Japanese government declared Japan to be ‘the world’s most advanced IT nation’ (Declaration to be the world’s most advanced IT nation, Cabinet Secretariat 2013). To this end, Japanese students will learn Informatics, and computer programming will be introduced even in elementary schools from 2020 (Kanemune *et al.*, 2017). In 2022, the curriculum guidelines for high schools will be drastically revised, with ‘Informatics I’ and ‘Informatics II’ having a predominantly scientific approach (MEXT, 2018), (Kano, 2017).

Given that a lot of ‘temporary teachers’ and ‘teachers without a proper license’ teach Informatics, and even most teachers who hold a proper license for Informatics are in

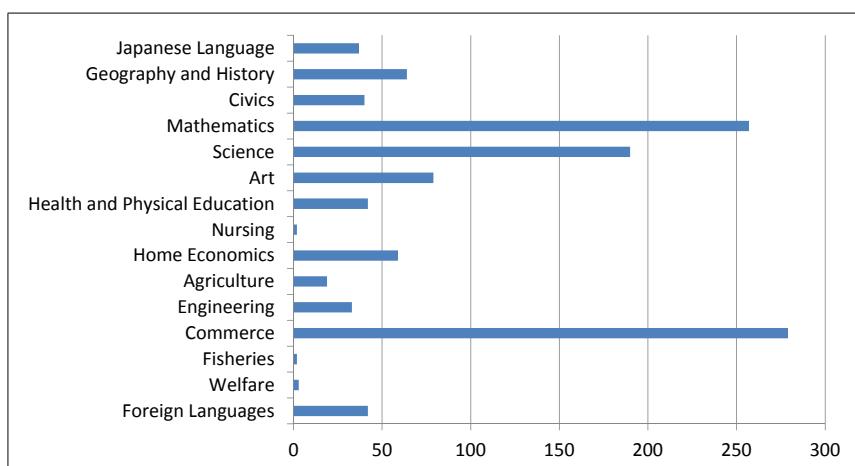


Fig. 2. Licensed subject for teachers who teach Informatics without proper license (May 2017).

charge of multiple subjects, it may be hard to implement new curriculum in dozens of prefectures that have few teachers who specialize in Informatics.

In order to enhance Informatics education, it will be necessary to develop laws like the ‘Informatics Education Promotion Act’. We believe that it is important to allocate a budget for and assign at least one teacher who specializes in Informatics to each school. It is important to employ teachers who specialize in subjects relevant to Informatics and education method in teacher training course at university to follow the remarkable ICT innovation.

References

- Cabinet Secretariat (2013). Declaration to be the world’s most advanced IT nation
http://japan.kantei.go.jp/policy/it/2013/0614_declaration.pdf
- Hagiya, M. (2015). Defining Informatics across Bun-kei and Ri-kei, *Journal of Information Processing*, 23(4), 525–530.
<http://doi.org/10.2197/ipsjjip.23.525>
- Hagiya, M. (2016). Disparity in Informatics education and reference standard in Informatics | Informatics as a basic discipline for Informatics education (in Japanese), *Jornal of Information Processing and Management*, 59(7), 472–478.
<http://doi.org/10.1241/johokanri.59.472>
- Kanemune, S., Shirai, S., Tani S. (2017). Informatics and programming education at primary and secondary schools in Japan, *Olympiads in Informatics*, 11, 143–150.
<http://doi.org/10.15388/ioi.2017.11>
- Kano, T. (2012). Current Situation of the subject of Informatics in Ishikawa prefecture (in Japanese), *JUCE Journal*, 138, 7–9.
http://www.juce.jp/LINK/journal/1203/pdf/02_02.pdf
- Kano, T. (2017). Revision of Course of Study and Common Subject Information Department (in Japanese), *IPSJ Magazine*, 58(7), 626–629.
<http://id.nii.ac.jp/1001/00182243/>
- MEXT (2018), Curriculum guideline for high school
http://www.mext.go.jp/a_menu/shotou/new-cs/1384661.htm
- Nakano, Y. (2006). A study of subject ‘Information’ in standpoint of board of education and teachers adoption (in Japanese), *IPSJ SIG Technical Reports*, 2006-CE-86-5.
<http://id.nii.ac.jp/1001/00054167/>
- Nakayama, Y., Nakano, Y., Kakuda, H., Kuno, Y., Suzuki, M., Wada, B.T., Hagiya, M., Kakehi, K. (2017). Current situation of teachers assigned for the subject of ‘Information’ at high-schools in Japan (in Japanese), *IPSJ transactions on computers and education*, 3(2), 41–51.
<http://id.nii.ac.jp/1001/00182185/>



Y. Nakayama is an associate professor in the Graduate School of Informatics and Engineering at The University of Electro-Communications. He received his B.E., M.E., and D.Eng. degrees from The University of Tokyo in 1988, 1990, and 1993, respectively. His research interests include operating systems, parallel and distributed computing, and Informatics education. He is a member of IPS Japan, and IEEE Computer Society.



Y. Nakano received his B.Eng., M.Eng. from Shibaura Institute of Technology in 1988, and 1990, respectively. He is a national certified Professional Engineer of Engineering Management and Information Engineering. He is a teacher in Kobe Municipal High School of Science and Technology. His research interests are on informatics education in secondary schools, teacher development and universities entrance examination.



Y. Kuno received the BS, MS, and PhD degrees in information science from the Tokyo Institute of Technology in 1979, 1981, and 1991, respectively. He is a professor in University of Electro-Communications, Tokyo, Japan. His current research interests include programming languages, programming education, and informatics education in general. He is a member of Information Processing Society of Japan, Japan Society for Software Science and Technology, Association for Computing Machinery, and IEEE Computer Society.



B.T. Wada is a Professor of Nagano University since 1984. He was a Visiting Professor of Division of Computer Education, College of Education, Korea University, Korea, in 2006. He received B.Eng in 1978 from Waseda Univ., MSc in 1982 from Univ. of Tsukuba. He acts as the Chairperson of Primary and Secondary education Committee, Informatics Education Committee, Information Processing Society of Japan. Senior member of IPSJ. Member of ACM. His current area of interest is Education of Informatics and its international comparative study.



H. Kakuda received his B.S., M.S., D.Science degrees from Tokyo Institute of Technology in 1974, 1976, and 1982, respectively. He was an associate professor in the University of Electro-Communications, and retired in March 2016. His research interests include human computer interaction, computers and education, Japanese document processing, and string manipulation. He is a member of IPSJ and ACM.



M. Hagiya received his B.Sc., M.Sc. from the University of Tokyo in 1980, and 1982, respectively, and his Ph.D. from Kyoto University in 1988. He is a professor at the University of Tokyo since 1995. His main research interests is in computational models, including those for molecular computing. He is also involved in activities for informatics education. He is a member of IPSJ and ACM.



K. Kakehi received his B.Eng., M.Eng. from the university of Tokyo in 1968, and 1970, respectively. He was a professor in Waseda University and retired in March 2016. His research interests is on programming, ranging over languages, tools, methods and education.

International School in Informatics “Junior” for IOI Training

Marina S. TSVETKOVA¹, Vladimir M. KIRYUKHIN²

¹*Academy of Natural History, Russian Federation
Russia, Moscow, 105037, box 47*

²*Dept. of Informatics and Control Processes, National Research Nuclear University
“MEPhI” 31 Kashirskoe Shosse, Moscow 115409, Russian Federation
e-mail: ms-tsv@mail.ru, vkiryukh@gmail.com*

Abstract. In 2018 the IOI will celebrate its thirtieth anniversary. Over these three decades, not only the world secondary school Olympiads in informatics community have been formed, which covers more than 80 countries from all continents, but a formation of an united methodological space of the school Informatics started also. This space allows many countries today to develop school computer science education, using the experience of other countries, materials from the IOI conference journal, sites of computer science contests, and other Internet resources. This article describes a model for organizing an international training event for juniors – International School in Informatics “Junior” – ISIJ.

Keywords: young talented children, informatics curriculum, model of IOI training.

1. Introduction

At the same time, the preparation of students in the different IOI countries is still very different. In order to attract more young people all over the world to participate in programming contest, and especially in IOI, it is necessary to solve two important tasks: to start training as soon as possible and to allow countries (children together with their coaches) to exchange experience (Kiryukhin, V. Tsvetkova, M., 2014).

2. Development of the International Olympiad Environment for Olympic Training of Juniors

IOI started in 1989 in Bulgaria, Europe, with the support of UNESCO. IOI has already been held all over the world. It has been repeatedly held in Europe and Asia. IOI 2016

was held in Russia placed in the two continents. IOI was held in the countries of North and South America, in Africa, and Australia. There is now no country that does not know about IOI, as the most important Olympiad in Informatics for secondary school students from all over the world.

The complexity of IOI tasks increases continuously and this requires strengthening of the coaches' work and responsibility. In order to have smoothed entering of the youngest programmers in the topics of IOI Curriculum the Olympic training should start from an early age. Only in such way children with interests in programming will have time to gain the necessary experience and to become competitive in IOI. Unfortunately, not all children in the countries of the world, that are 12–15 years old, have a school Informatics courses at this age. Something more, not all teachers of these children can become coaches, because there are different curriculums of school Informatics in different countries. One possible way for having an efficient training in Informatics for juniors is to create an open teaching environment of Olympic school Informatics for junior age and to integrate as many as possible children of 12–15 years in it.

Due to the emphasis on the early training of juniors in Olympic Informatics, the process of searching for ways of methodological integration of IOI member countries for work with talented children aged 12–15 has begun. Since 2015, IOI team leaders have been discussing the formation of an international Olympiad environment for training juniors. At IOI 2016 the authors presented the Russian experience in preparation of juniors for olympiads in informatics and experience of Kazan Federal University (KFU) in creating in Russia of regional IT educational infrastructure as “digital bridge” from school to university (Tsvetkova M., 2016). After that in 2017 Bulgaria started a new programming contest – the European Junior Olympiad in Informatics (eJOI, 2017) – a possibility for more young programmers to take part in a serious international competition.

In support of these initiatives, Russia has offered to organize an international school in Informatics for juniors that are preparing for participation in international competitions in Informatics – International School in Informatics “Junior” (shortly ISIJ or the School). The mission of ISIJ is to create a space of Olympic startup in Informatics for all juniors of the world, to unite school teachers-innovators in Informatics in international coaching community, to stimulate exchanging of experience and creation of tasks for talented juniors. It is important that in ISIJ the coaches of the juniors will participate too. They will cooperate in the International Coaching Council, will exchange experience, will create tasks, and will receive methodical training for work with young talents. In such way ISIJ will be a training place for young IT specialists and coaches too.

The School is organized at the site of KFU, which hosted IOI 2016. The first session of the School was held in the summer holidays of 2018 for children from Europe and Asia, the second one is planned for the winter holidays for other interested countries.

We hope that ISIJ will become an attractive event for Olympic training in Informatics of juniors from different countries of the world and their coaches.

3. Organization Structure ISIJ

ISIJ is organized by KFU together with scientists from the national Olympic teams in Informatics of Russia and Bulgaria. The governing body of the School in Russia is the Organizing Committee, which is formed by the KFU administration and scientific consultants from the Russian Olympic team in Informatics.

The program and the financial support of the School are supervised by the Executive Directorate of the School in coordination with the Organizing Committee. The School Directorate is headed by an Executive Director who organizes the registration of participating countries, registration of teams, implementation of the School's program, and manages the School Fund. The Executive Director attracts a team of specialists in the Program Secretariat of the School. He organizes information mailing, collection of participation fees, and visa support for foreign teams.

Information about the time schedule of the School and the participation fees is specified in the letter of the Organizing Committee to the coaches of the teams of the registered participating counties. The participation fees includes only the accommodation and the meals for participants (breakfast, lunch, dinner and buffets), as well as some cultural program.

In the residence of participants there is a medical staff and supporting teachers from IT lyceum (KFU, 2018). A security measures for participants are provided. The residence of the School is in a children's educational institution and has comfortable conditions for living and conducting classes. All rooms are equipped with computers and Internet.

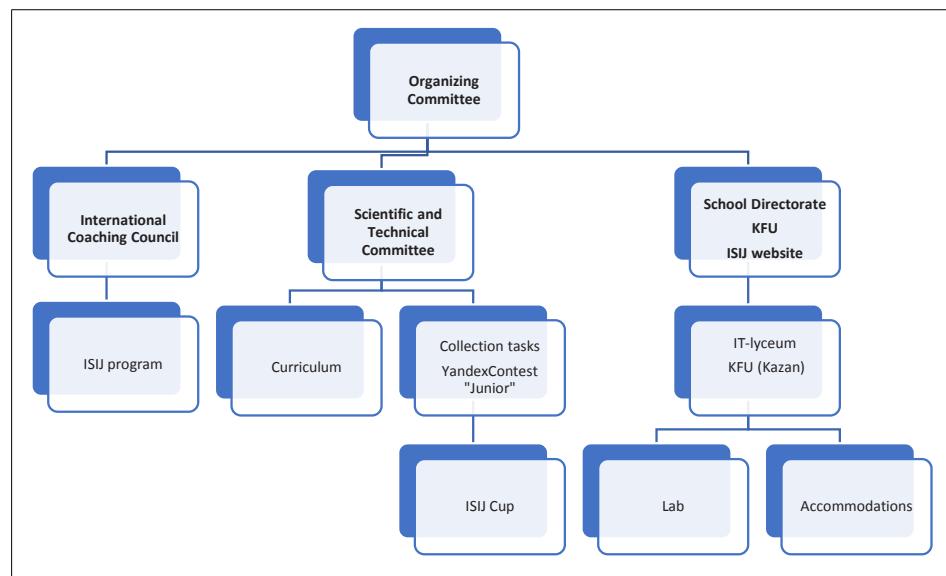


Fig. 1. Organization structure ISIJ.

4. Participants of ISIJ

The School is held as an international educational start-up for the national teams of participating in IOI countries: Summer school – for the countries of Europe and Asia, Winter school – for other interested countries.

The team of a country consists of 2 to 6 participants and one coach – the Head of the team. Age of participants: at least 12 years of age and not older than 15 years at the end of the previous year. Children from Russia that will participate in the School will be contestants from the national Olympiad, and will be invited by the administration of the School.

5. Scientific and Technical Committee – STC ISIJ

Scientific and technical support of the School's activities is provided by the Scientific and Technical Committee (STC). School's Organizing Committee is composed in coordination with the School STC.

Technical support of the School is carried out by the sponsor of the School – the Russian IT company Yandex – on the basis of a cloud system for competitions organizing and grading (Yandex Contest "Junior"), specially made for the School and opened for filling with tasks by the Coaching Council and STC of the School.

The composition of the Russian group of member of STC is approved by the Organizing Committee of the School. Applications for membership of coaches from the different countries in the international part of the STC are made by the countries. STC of the School is headed by two leaders – one from Russian members and the second one – from the international STC members. Together, both leaders organize the work of the Coaching Council to fill the collection of tasks.

Preparation of the training tournaments and the ISIJ Cup tournament is carried out by the STC of the School using tasks from the national Olympiads of Bulgaria, Russia and other participating countries. STC start preparation of tournaments tasks in the Yandex Contest system (remotely) one year before the start of the next School. The participating countries can form in the Contest system their own sections of tasks useful for their teams during the School and for remote tournaments in the period between two School's sessions.

Tasks from the tournaments of the current year, prepared by the Coaching Council of the School are available for participants registered for the next School as remote homework.

6. The International Couching Council

Coaches of the participating countries compose the International Coaching Council (ICC) of the School: one coach from each participating country.

The representative of each country in ICC provides every year one ready for use task for the collection of tasks of the School. The task archive has to contain: statement of

the task – in Russian or English, program-solution and checker (if necessary) written in C++ programming language, and test cases. Description of the alternative possible solutions is welcome.

The ICC chooses some of its members to represent ICC in the international group of STC by public vote. The term of representation in the international group of STC is limited to 5 years.

The Coaching Council of the School participates actively in the work with juniors – giving consultations after tours, organizing workshops, providing translation of the tasks to the national languages, holding thematic meetings for selection and refinement of tasks for the task collection of the School.

7. ISIJ Program

The School is held for 11 days in the summer and for 7 days in the winter. The School’s program includes at least five training tournaments and one competitive tournament – the ISIJ Cup. STC and the Coaching Council decide on the educational content of the tasks on the basis of the Curriculum of EJOI (eJOI, 2017) and then select set of appropriate tasks (according to Curriculum), on the basis of which the STC prepares tournaments. STC and Coaching Council prepare a set of lessons to be proposed to participants and organize pedagogical workshops and consultations for groups of coaches.

The School’s languages are Russian and English. The organizers provide participants with technical devices for simultaneous translation for small groups with a common language. The texts of the tasks are provided to all participants in Russian or English in electronic form with the possibility of printing. The coaches could provide the participants with a translation of the tasks into the native language of the students.

The program includes two excursions. The program provides sports, leisure activities, chess classes, meeting with people from IT companies, scientific and practical workshops.

The Organizing Committee approves the terms of the School taking into account the proposals of the ICC, conducts meetings with the participants, and manages the provision of a base for the School in the campus of KFU.

8. ISIJ – Cup

On the last day of the school a ISIJ-Cup is held. The Cup is held in one round of 3 tasks for 4 hours.

Following the results of the Cup three places for rewarding are defined: gold, silver and bronze medalists of the Cup. In case of equal number of points, all participants with these points are awarded the medal.

From the Russian group of participants no more than 6 participants (chosen by their performance rating from tournaments of the School) are allowed to compete in the Cup

are, i.e. the Russian team is has no more members than the teams of the other participating in the School countries.

The Coaching Council determines the scores necessary for awarded the Diplomas of the Cup finalists. The number of the Diplomas is no more than 1/2 of the number of the Cup participants.

9. ISIJ Website

On the School's website (ISIJ, 2018) all teaching materials of the School are published and they are accessible for the registered participants of the School only.

The site of the School contains information about the composition and contacts of the Organizing Committee, the STC, the Directorate, and the Coaching Council, a list of participating countries, the program of the School, the schedule of the Coaching Council, School news, etc.

The archive of tasks of the School is placed in a special section on the Website of the school in order to be used by registered participants and coaches through the Yandex-Contest system for the remote access throughout the year. Registration of all participants on the school website is mandatory.

The site contains the logo of the School for the use by countries in the design of t-shirts and other objects attributes of the School.

On the website of the School there is a history page in the form of a virtual whiteboard achievements of school participants: the best participants of the year, medalists and finalists of the school Cup, etc.

10. Certificates of Participation

All participants of the International School receive a certificate of participation.

All school coaches receive a certificate of teaching internship in the amount of 72 hours under the program "Methods of development of early talent in informatics". (Tsveikova M., 2010).

Ten best participants of ISIJ on the sum of all training rounds receive the diploma with honors and information on them is placed on a virtual distinction Board on the website of school.

11. Conclusion

The organizers of the ISIJ hope to form an international school for juniors, including all countries participating in IOI as well as to provide juniors from these countries remote participation in the training tournaments of the School.

The total experience of the IOI coaches in the implementation of the ISIJ (as the start of the Olympic "lift") will form the best methods of developing the talent of the juniors and methods of teaching informatics in schools in the world. The ISIJ will be an open Olympiad educational space for juniors in informatics.

References

- eJOI (2017). *European Junior Olympiad in Informatics*. <http://ejoi.org/>
ISIJ (2018). *International school in informatics "Junior"*.
<http://www.isij.universityevents.ru/isij2018>
KFU (2018). *IT-lyceum of Kazan Federal University*. <https://kpfu.ru/it-liceum/infrastruktura/>
Kiryukhin, V.M., Tsvetkova, M.S. (2010). Strategy for ICT skills teachers and informatics olympiad coaches development. *Olympiads in Informatics*, 4, 30–51.
Kiryukhin, V.M., Tsvetkova, M.S. (2014). The approach of early olympiad preparation "Olympic Lift". *Olympiads in Informatics*, 8, 111–122.
Tsvetkova, M.S. (2016). Foreword. *Olympiads in Informatics*, 10 (special issue), 1–2.



M.S. Tsvetkova is professor of the Russian Academy of Natural Sciences, PhD in pedagogical science, prize-winner of the competition "Teacher of the Year of Moscow" (1998). Since 2002 she is a member of the Central methodical commission of the Russian Olympiad in informatics, the pedagogic coach of the Russian team for IOI. She is author of many papers and books in Russia on the digitalization of education and methods of development of talented students. Since 2013 she is the Leader of the Russian team for IOI. Since 2017 she is Expert of the Committee on Education and Science of the Parliament (State Duma) of the Russian Federation.



V.M. Kiryukhin is professor of the Russian Academy of Natural Sciences and assistant professor of National Research Nuclear University "MEPhI". He is the Chairman of the federal methodical commission in Informatics, which is responsible in Russia for carrying out the national Olympiad in Informatics. He is author of many papers and books on development of the Olympiad in Informatics in Russia and preparations for the Olympiads in Informatics. He is the exclusive representative who took part at all IOI from 1989 as a member of the IC of IOI (1989–1992, 1999–2002, 2013–2016) and as a Leader of the Russian team (1989, 1993–1998, 2003–2012). He received the IOI Distinguished Service Award at 2003 and 2008 as one of the founders of the IOI and for his long term distinguished service to the IOI from 1989 to 2008, as well as the medal "20 Years International Olympiad in Informatics" at 2009.

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 journal consists of two sections: the main part is devoted to research papers and only original high-quality scientific papers are accepted; the second section is for countries reports on national olympiads or contests, book reviews, comments on tasks solutions and other initiatives in connection with teaching informatics in schools.

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
- 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:

- International Olympiads in Informatics* (2008).
<http://www.IOIinformatics.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 Dagiènè
Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
Phone: +370 5 2109 732
Fax: +370 52 729 209
E-mail: valentina.dagiene@mii.vu.lt

Internet Address

All the information about the journal can be found at:

http://ioinformatics.org/oi_index.shtml

Publisher office: Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
September, 2018

Olympiads in Informatics

Volume 12, 2018

T. BELL	Computer Science in K-12 Education: The Big Picture	3
M. DOLINSKY, M. DOLINSKAYA	How to Start Teaching Programming at Primary School	13
M.C. FONTAINE	Tidal Flow: A Fast and Teachable Maximum Flow Algorithm	25
D. GINAT	Algorithmic Cognition and Pencil-Paper Tasks	43
M. JOVANOV, M. MIHOVA, B. KOSTADINOV, E. STANKOV	New Approach for Comparison of Countries' Achievements in Science Olympiads	53
T. KAKESHITA	National Survey of Japanese Universities on Computing Education: Analysis of Departments Majored in Computing Discipline	69
B. KOSTADINOV, M. JOVANOV, E. STANKOV	Platform for Analysing and Encouraging Student Activity on Contest and E-learning Systems	85
H. MANABE, S. TANI, S. KANEMUNE, Y. MANABE	Creating the Original Bebras Tasks by High School Students	99
P.S. PANKOV, A.A. KENZHALIEV	Combinatorial Property of Sets of Boxes in Multidimensional Euclidean Spaces and Theorems in Olympiad Tasks	111
W. van der VEGT	How Hard Will this Task Be? Developments in Analyzing and Predicting Question Difficulty in the Bebras Challenge	119
REPORTS		
N. AMAROLI, G. AUDRITO, L. LAURA	Fostering Informatics Education through Teams Olympiad	133
M. ANDERLE	PRASK – an Algorithmic Competition for Middle Schoolers in Slovakia	147
Á. ERDŐSNÉ NÉMETH, L. ZSAKÓ	Grading Systems for Algorithmic Contest	159
Y. NAKANO, K. IZUTSU	The Next Course of Study from 2022 and a History of the Subject "Informatics" in Japanese High Schools	167
Y. NAKAYAMA, Y. NAKANO, Y. KUNO, B.T. WADA, H. KAKUDA, M. HAGIYA, K. KAKEHI	Current Situation of Teachers of Informatics at High Schools in Japan	177
M.S. TSVETKOVA, V.M. KIRYUKHNN	International School in Informatics "Junior" for IOI Training	187