Olympiads
in Informatics

14

OLYMPIADS IN INFORMATICS

Selected papers of
the International Conference joint with
the XXXII International Olympiad in Informatics
(online) Singapore, 13–19 September, 2020

OLYMPIADS IN INFORMATICS

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

```
https://ioinformatics.org/page/ioi-journal
```

Foreword

IOI, the International Olympiad in Informatics, organized by Singapore from September 13th to 19th, 2020, is held online for the first time in the IOI history, due to the worldwide spread of COVID-19. This is a big challenge for host organizers, as well as IOI committees. A traditional one-day scientific conference is replaced by only virtual presentations of the papers by the authors, and publishing papers in this volume in printed and online version at IOI website. The authors are welcome to make short onsite presentations at next year IOI, which will be hosted by Singapore onsite, 20–27 of June, 2021.

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. Unfortunately, the conference this year will be held as virtual event with online presentations from the authors. The 14th volume has two tracks: the first section of the journal focuses on research, and the second section includes reports for sharing national experiences and other news important for the community.

In their paper, A. Alnahhas and N. Mourtada study the possibility of using machine learning techniques in order to build a system that will be able to predict the future performance of contestants in competitive programming contests, based on historical rating lists. They describe an experiment in which they use public data from the Codeforces website and show that machine learning techniques achieve acceptable results for the problem.

P.T. Do, B.T. Pham and V.C. Than summarize the latest algorithms for solving some classical NP-hard optimization problems on the particular graph classes. Furthermore, they discuss the application of the α–redundant method in order to obtain linear-time algorithms for finding a Maximum Induced Matching on interval and circular-arc graphs.

M. Dolinsky and M. Dolinskaya describe a system for teaching text-based programming in elementary schools, built on the basis of the website dl.gsu.by, and explain the main advantages of using this technology.

The paper of D.I. Estevez introduces theory of blockchain technology, implementation, and applications while focusing on its types of consensus algorithms. The methodology is linguistic and consists of a comparative analysis of the most popular algorithms. The problems related to blockchain architecture and algorithms could serve as an inspiration and a training resource for olympiads because of their potential to be intellectually stimulating and to contribute to our knowledge.

In their paper „Recommending Tasks in Online Judges using Autoencoder Neural Networks", P. Fantozzi, and L. Laura propose the design of a recommender system for tasks in Online Judges. The authors think that their system is a first step to the development of recommender systems.

In his paper "Operator Utilization and Abstract Conceptions", D. Ginat deals with abstraction. The abstract perspectives of ignoring details and relating to particular properties of recognizable parts are fundamental in problem solving. The task solutions presented in the paper encapsulated them in computations with repeated utilizations of operators. Two operators are used for sorting and one is used for transforming binary matrix colors.

In the paper "Introduction of Honorable Mention award at International Olympiad in Informatics", M. Jovanov and E. Stankov present analysis of the results at IOI and other Scientific Olympiads in previous years, and present a final solution for the introduction of the fourth level award at IOI.

A. Laaksonen and T. Talvitie present the current state and future plans for the CSES (Code Submission Evaluation System) online judge system, which has been used for organization of several online programming courses as well as contests in Finland since 2013. The CSES problem set is an ongoing project whose purpose is to create a high-quality collection of educational algorithm programming problems. At the time of publishing the problem set consists of 200 problems, and the final goal is to reach 1,000 problems.

M. Lodi analyses the concept of computational thinking by providing a review of its many definitions and finding that they share a lot of common elements, classified in mental processes, methods, practices, and transversal skills. He concludes that elements of computational thinking should be intended inside Informatics as a discipline, being its "disciplinary way or thinking" – Informatical thinking.

A group of authors (M. Mirzayanov, O. Pavlova, P. Mavrin, R. Melnikov, A. Plotnikov, V. Parfenov, and A. Stankevich) discuss Codeforces as an educational platform for learning programming. They argue that the infrastructure of Codeforces provides a solid open ecosystem for building a programming learning process, and describe all the aspects and relationships of this ecosystem, as well as examples of successful integration into educational processes in the age of digitalization.

A paper of P.S. Pankov and A.A. Kenzhaliev deals with tasks on recognition and on restoration of data from a unified viewpoint.

M.S. Tsvetkova and V.M. Kiryukhin elaborate top 10 key skills which they have concluded to be most valuable for the success of students at IOI, based on their 30 years long experience on preparation of IOI gold medallist in Russia, as well as on preparation of curriculum of school informatics.

Finally, in the second part of the volume, two reports are presented focusing on team selection and training in Hungary, and in Argentine Olympiad in informatics. S. Halim presents the latest edition of his book aimed at all potential IOI contestants around the world, for their preparation for the contest. B. Kostadinov gives report of a new IOI activity aimed at the period in between two actual IOIs, that will keep the interest and spirit of the involved people from IOI community.

Many thanks to all of those who have assisted with the volume – especially authors and reviewers. A lot of work goes, not only to the writing of the papers, but to an extended period of review and correction.

<div align="right">Editors</div>

Predicting the Performance of Contestants in Competitive Programming Using Machine Learning Techniques

Ammar ALNAHHAS, Nour MOURTADA
Faculty of information technology engineering
Damascus University – Damascus – Syria
e-mail: eng.a.alnahhas@gmail.com, nourmourtadaite@gmail.com

Abstract. Training is an important task in competitive programming, coaches can improve the training experience if they can get information about the performance of contestants. In this paper, we study the possibility of using machine learning techniques to build a system that is able to predict the future performance of a contestant by analyzing their historical rating list. This system learns from a dataset of contestant ratings. We propose to apply five different baseline machine learning techniques, then we propose a new deep learning model. We conduct an experiment using public data from the Codeforces website. We show that most techniques achieve acceptable results. In addition, the proposed deep learning model outperformed all baseline methods and achieved results that proved its efficiency in predicting the future performance of contestants. This paper confirms the possibility of using machine learning techniques to help in the process of preparing contestants in competitive programming.

Keywords: competitive programming, machine learning, artificial intelligence, programming training, Informatics Olympiads.

1. Introduction

Competitive programming has become very popular in recent years. It has been attracting more interest all over the world. Many important competitions in competitive programming are organized each year, such as the International Olympiad in Informatics (IOI) and the International Collegiate Programming Contest (ICPC). Thousands of contestants are participating in competitive programming competitions that are held online and onsite all over the world.

Training is a key element in preparing for competitive programming, therefore, many institutions like organizers of national Olympiads in informatics and universities are interested in creating and adopting successful training plans for their contestants (Combéfis & Paques, 2015), and many training methods are introduced and discussed.

Because of the importance of the training, many coaches follow up with their contestants to support their training process and provide suitable materials and tools. For this reason, it is important for coaches and people in charge of training to observe and track contestant performance. It is useful if they have an indicator if the performance of a contestant is decreasing, so they can help in the early stage and support the contestant.

Artificial intelligence has gained more interest in both research and application in the recent years, it is becoming part of everyday life, many intelligent applications are helping people doing their work, and it has been used in modern educational systems to make them more adaptive for learners (Colchester, Hagras, Alghazzawi, & Aldabbagh, 2017), machine learning is one of the most interesting branches of artificial intelligence, this field of science is interested in building intelligent systems that can learn by itself, usually by observing and analyzing a large amount of available data.

As thousands of contestants are now interested in competitive programming, online training websites like Codeforces provide large amounts of contestant data, so it is possible to build an intelligent system to help analyze this data and support the training process using artificial intelligence.

In this paper, we are going to present a methodology that aims at using machine learning techniques in order to build a system that can predict the future performance of competitive programming contestants by analyzing a sequence of their historical ratings. We implement some known machine learning models, then we propose a new deep learning model and prove its efficacy in tracking contestants' performance by providing results of empirical experiments on data from Codeforces.

Although coaches can observe their contestants to ensure their performance is not going to decrease, there are some reasons an automatic system can support this process:

- The relation between contestant performance and their historical ratings may be not simple to be detected by humans. Complex patterns may exist in this case. Computer systems can detect these patterns especially using machine learning techniques by analyzing big amount of data, and extracting useful information by generalization.
- The number of contestants may be large for coaches to follow up in some cases, so an automatic system can help by pointing the coaches to potential performance-decreasing contestants, so that they can do further observations and check the situation.

Therefore, the proposed system can be seen as a decision support system that helps coaches during the training process by providing an early alarm, so that they can act in a timely manner. This system can help coaches of national Olympiads where number of contestants participating in the training process may be large.

This paper is organized as follows: section 2 provides some related works in the field of predicting student performance, section 3 presents a formal description of the problem we are trying to solve, section 4 introduces the baseline machine learning models application, section 5 proposes a new deep learning model, section 6 contains the details and results of the experiments, and finally section 7 concludes this paper.

2. Related Work

Although no researchers have tried to predict the performance of contestants in competitive programming using artificial intelligence techniques; much research has focused on using related methodologies to predict school or university student performance as well as detecting students with poor educational progress. This research varied in the way they addressed and solved the problem. Some of them used traditional machine learning techniques, while others tried to use deep learning methods such as convolutional and recurrent neural networks.

Kotsiantis, Pierrakeas, & Pintelas (2004) proposed to use various machine learning algorithms to predict the performance of students in a distance learning environment, they used a dataset that consists of students in an informatic course. The students were represented by two types of attributes: demographic and performance. Five different machine learning techniques were applied to predict if a student would succeed or fail in the final exam. The results showed that the Naïve Bayes algorithm achieved the best results. Tanner & Toivonen (2010) tried to predict students' final exam results in an online touch-typing course in which each student had to pass 12 lessons and a final exam. The goal of the research was to predict in early stages of the course if a student was going to fail the exam so that the teachers could provide more tutoring. The researchers suggested using the K-nearest neighbors (KNN) algorithm, a machine learning technique that depends on the idea that students with similar attributes tend to have similar exam results. The authors reported good results using KNN in predicting student performance. Tan & Shao (2015) presented a machine learning-based system that helped predict student dropout in an educational system. They argued that the performance of the students is an important factor of dropout, so they suggested building a binary classifier that can predict if a student will drop out or graduate eventually. The proposed system used the grades of each student as input, and applied two different algorithms to analyze the data: logistic regression and decision trees. The authors showed that both methods achieved good results. Amra & Maghari (2017) proposed a system that can predict secondary students' future performance based on various attributes. They compared two different machine learning algorithms: K-nearest neighbors and the Naïve Bayes classifier. The results they presented showed that the Naïve Bayes model achieved higher accuracy. Babić (2017) presented a system that aimed at predicting student academic motivation. The proposed system depended on the behavior of students in an online learning management system, and used three different machine learning methods to classify students: artificial neural networks (ANN), decision trees and support vector machines. The results were promising and could help educators find students with poor performance at early stages. Waheed *et al.* (2020) tried to build a system that can predict the academic performance of students in a virtual learning environment based on clickstream data and assessment results. Their system used artificial neural networks to categorize students into two classes: success and failure. Authors compared the results with baseline methods: logistic regression and support vector machines, and proved that ANN outperformed them. Similarly, researchers Hussain *et al.* (2019) proposed to use artificial neural networks to predict student performance based

on internal assessment results. They presented the issue as a classification problem, as other researchers did, but they compared the results of ANN with two different classifiers: The Artificial Immune Recognition System and AdaBoost. The proposed model used exam scores and internal assessment marks; the results showed that ANN outperformed other methods. Sekeroglu, Dimililer, & Tuncal (2019) also used neural networks to predict student performance. They suggested two different neural networks, multilayer perceptron (MLP) and recurrent neural networks, and compared the performance of these two networks with support vector machines. MLP proved to be best at classifying students and predicting performance. Koutina & Kermanidis (2011) tried to build a system to predict the performance of postgraduate students in order to help tutors detect students at risk of failing in early stages. The authors used students' marks along with some demographic attributes to predict the performance and compared six well-known machine learning algorithms including support vector machines, Naïve Bayes and K-nearest neighbor. Their research led to the result that the Naive Bayes classifier achieved the best results.

Many other researchers were interested in this field (Al-Shabandar *et al.*, 2017; Ofori, Maina, & Gitonga, 2020; Thai-Nghe, Drumond, Krohn-Grimberghe, & Schmidt-Thieme, 2010; Xu, Moon, & Van Der Schaar, 2017). The successful use of machine learning and deep learning techniques to predict the performance of students in the previous work is a good indicator that these methods can help predict the performance of contestants in competitive programming. Especially as many researchers use a similar input pattern which is the sequence of level rates of the contestant. We are going to apply different well-known traditional machine learning techniques along with a new deep learning model to try to achieve the research goal.

3. Formal Problem Statement

The problem we present in this paper is about predicting the future performance of a contestant participating in a competitive programming training program. Each contestant is assumed to have a rate that reflects their excellence level, which changes over time to reflect change in contestant performance. The goal is to predict if the level of the contestant is going to increase or decrease according to the historical ratings we already recorded for them.

We can measure the performance of a contestant by the average of their level ratings, so if the average increases over time, then we can say that the student performance is getting better and vice versa.

Formally, if a contestant c has a sequence of n temporally ordered ratings $R = r_1, r_2, ..., r_n$, we define the function $RC(R)$ as the following:

$$RC(R) = \frac{1}{n/2} \sum_{i=n/2}^{i=n} r_i - \frac{1}{n/2} \sum_{i=1}^{n/2-1} r_i$$

The function RC denotes the difference in rating average between the first half and the second half of the rating sequence. Intuitively, if the function RC has a positive value, the contestant performance is increasing while a negative value means the performance is decreasing, and this is the case we are interested in, because detecting contestants who tend to do worse in the future will guide the coaches to provide suitable solutions. Formally, we define the function $T(R)$ as the following:

$$T(R) = \begin{cases} 1, & RC(R) < 0 \\ 0, & RC(R) \geq 0 \end{cases}$$

So, the goal of the models in this paper is to predict the value of $T(R)$ given $n/2$ ratings of a contestant. As the value of $T(R)$ is binary, the problem can be addressed as a binary classification, that is, each sequence of ratings belongs to either a positive or negative class. Binary classification problems are well-known to be solved by many machine learning and deep learning methods that can be trained using a dataset of existing instances, so the model can generalize and classify new instances of data.

To solve the proposed classification problem, we will apply different types of baseline machine learning algorithms, then present a novel deep learning model that achieves good results and outperforms all other methods, as we will elaborate in the results section.

4. Baseline Machine Learning Algorithms

Machine learning is used to discover data patterns and relationships between variables, which is helpful in the decision-making process. It is a useful tool for detecting contestants whose level is going to decrease based on their rating sequence. Coaches are then able to help the weakest ones in a timely manner, and to promote the strongest, thereby improving contestants' level. There are several well-known machine learning models that are usually used in classification tasks. We choose five of them to apply in our study, and will provide a brief description of how we apply them in this section.

4.1. *Random Forest*

Random forest (Liaw & Wiener, 2002) is a classification methodology widely used in machine learning. Random forest is a collection of decision trees (Quinlan, 1986) built up with some element of random choice. Each decision tree is constructed using random features of the data. The trees are not pruned, and each leaf of each tree represents a class. The algorithm is trained using a bootstrap sampling method. The prediction of a new item is done by the voting technique, that is, the prediction of each tree is found according to the leaf reached, then each class is assigned a ratio of trees that voted for it. The random forest algorithm has been successfully used in many applications; in our

case we construct the forest using our dataset. We build trees from the ratings of the contestants, and the forest we construct contains 100 decision trees. We use Gini impurity as a criterion for splitting tree nodes.

4.2. *Logistic Regression*

Logistic regression is a binary classification model. It is considered as a baseline for any binary classification problem, and is a basic fundamental concept in machine learning. It describes data and estimates the relation between one dependent binary variable and independent variables. It is a special case of linear regression where the dependent binary variable is categorical in nature. Linear regression gives a continuous output, but logistic regression gives a constant output. Logistic regression is suitable for our goal, where the dependent variable is the contestant class or performance prediction, and the independent variables are the historical ratings of this contestant. Given the rating sequence $R = r_1, r_2, ..., r_n$ of a contestant, we define a linear function Z:

$$Z(R) = \beta_0 + \beta_1 r_1 + \beta_2 r_2 + \cdots + \beta_n r_n$$

The factors $\beta_0, \beta_1, \beta_2, ..., \beta_n$ are the model parameters whose value should be found by the model fitting process.

We apply a logistic function L to the result of the above function to get the value in the range (0,1):

$$L(R) = \frac{1}{1 - e^{-Z(R)}}$$

The function L represents the output of the model, and the predicted class is then found by choosing a suitable threshold th and find the class accordingly:

$$class = \begin{cases} 0, & L(R) < th \\ 1, & L(R) \geq th \end{cases}$$

The training of the model is done by minimizing the mean squared error between the function L of the model and the real output of the training samples.

4.3. *Artificial Neural Networks*

A neural network consists of connected items called neurons, where each neuron has multiple inputs and a single output. There are many types of neural networks which vary in the way the neurons are connected to each other. Multilayer perceptron (MLP) is one of the famous types of neural networks. In MLP neurons are arranged in consecutive layers where the output of neurons of each layer constitutes the input of the next layer.

The first layer is the input layer, and the last layer is the output layer whose output is considered the output of the whole network. MLP is used widely in machine learning as it can learn from existing samples of data and generalize the pattern to be applied to new items, so it is useful in various classification and regression tasks. In our work, we use MLP as a binary classifier; we use a three-layer MLP. The first layer is the input layer which consists of neurons where each neuron corresponds to a rating from the input, the second layer is a hidden layer and the third layer is the output layer which consists of a single neuron. Fig. 1 shows the network architecture. The activation function for the first and second layers is the ReLU function, which is a positive linear function, whereas the activation function of the output layer is sigmoid:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

The output of the Sigmoid function is in range (0,1). To get the final class we choose a suitable threshold *th* and find the class accordingly:

$$class = \begin{cases} 0, & output < th \\ 1, & output \geq th \end{cases}$$

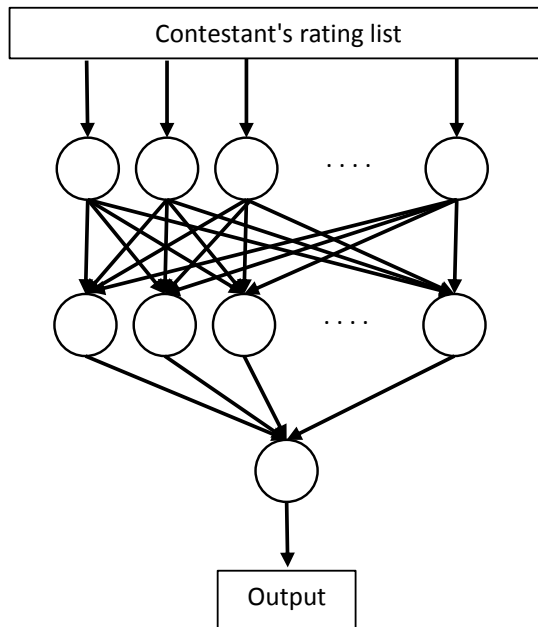To train the network we use the Adam algorithm (Kingma & Ba, 2014) and cross entropy as a loss function.

Fig. 1. Proposed MLP architecture.

4.4. *Naïve Bayes Classifier*

The Naive Bayes algorithm is an effective and efficient classification method in machine learning. A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. Bayes' theorem depends on the following relationship, given class variable y which denotes contestant performance prediction and the dependent feature vector, which is the historical ratings of this contestant r_1 through r_n:

$$P(r, \dots, r_n) = \frac{P(y)P(r_1, \dots, r_n|y)}{P(r_1, \dots, r_n)}$$

To train our model, we use two different methods. Firstly, we train a **Gaussian Naïve Bayes Model** which implements the Gaussian Naive Bayes algorithm for classification. Gaussian Naïve Bayes theorem states the following relationship:

$$P(y) = \frac{1}{\sqrt{2\pi\sigma_y{}^2}} \exp\left(-\frac{(r_i - \mu_y)^2}{2\sigma_y^2} \right)$$

Secondly, we train a **Multinomial Naïve Bayes Model** which implements the naive Bayes algorithm for multinomial distributed data. The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \theta_{y2}, \dots, \quad \theta_{yn})$ for each class y.
Where:

- n is the number of ratings.
- θ_{yi} is the probability $P(r|y)$ of rating i appearing in a sample belonging to class y.

Multinomial Naïve Bayes theorem states the following relationship:

$$\theta_{y_i} = \frac{N_{yi} + \propto}{N_y + \propto n}$$

Where:

- $N_{yi} = \sum_{r \in T} r_i$ is the number of times rating $i\ i$ appears in a sample of class yy in the training set TT.
- $N_y = \sum_{i=1}^{n} N_{yi}$ the total count of all ratings for class yy.

4.5. *Support Vector Machine*

The Support Vector Machine (SVM) is one of the most popular machine learning classifiers. It is a supervised learning algorithm used for both classification and regression. SVM is effective and efficient for two-class problems; it is based on finding a hyper-

plane in n dimensional space that splits the data points. This hyperplane should have the maximum margin to the nearest data points, and it forms a decision boundary so that the separation between two classes is as wide as possible. In this paper, as our target is formulated as a binary classification problem, we train a Support Vector Machine Classification Model in the space with n dimensions where n is the number of historical ratings we consider for each contestant. We train a nonlinear support vector machine with the kernel trick, using the radial basis function kernel.

As SVM output is discrete, probabilistic Support Vector Machine (PSVM) is a variation of SVM where multiple SVM is applied with multiple cross-fold operations on the dataset, the output of PSVM is continuous and falls in range [0,1]. In this paper we try out both SVM and PSVM.

5. Proposed Deep Learning Model

Deep learning is a subclass of machine learning. In deep learning, data itself is not considered as features for classification, but higher-level features are extracted from the data in order to enhance the accuracy of the classification process by finding features that are conceptually more expressive.

Deep neural networks are the most common implementation of deep learning, where some layers are added to extract semantic representation of data that is used for classification with ANN layers.

Recurrent neural networks (RNN) (Jain & Medsker, 1999) are well-known for processing sequential data, especially when the data is temporally ordered. This network consists of units, where each unit corresponds to an item of the processed sequence. The input of each unit is formed by both the output of the previous unit and the corresponding item from the sequence; hence, the network is able to capture the temporal relation between the items of the data sequence. Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) is a special type of RNN which is more suitable for processing longer sequences.

LSTM can extract latent vector representation, called embedding, from a data sequence; the embedding can represent data in a different (usually lower) dimension and with some useful semantics.

The model we propose in this paper is a deep neural network that consists of five layers: an embedding layer followed by two LSTM layers and finally two ANN layers. The ratings of the contestants are integers that reflect their levels. These integers are not sufficient to be used as the input to recurrent networks, so the role of the embedding layer is to find a vector representation of these ratings expanding the dimensionality of data from 1 to 64 (we chose to have the layer generate a vector of 64 values for each rating value). The output of the embedding layer is n vectors, each with 64 values. This output goes to the second layer, which is an LSTM layer consisting of 16 units. Each sequence item is used as an encoder which transforms each vector of its input from 64 values into 16 values, decreasing the sparsity of the embedding representation. To achieve this, the output of this unit is formed by the output of each unit, that is, the output is n vectors each having 64 values. The output is passed to the next LSTM layer which contains 16 units for each item as well; this layer
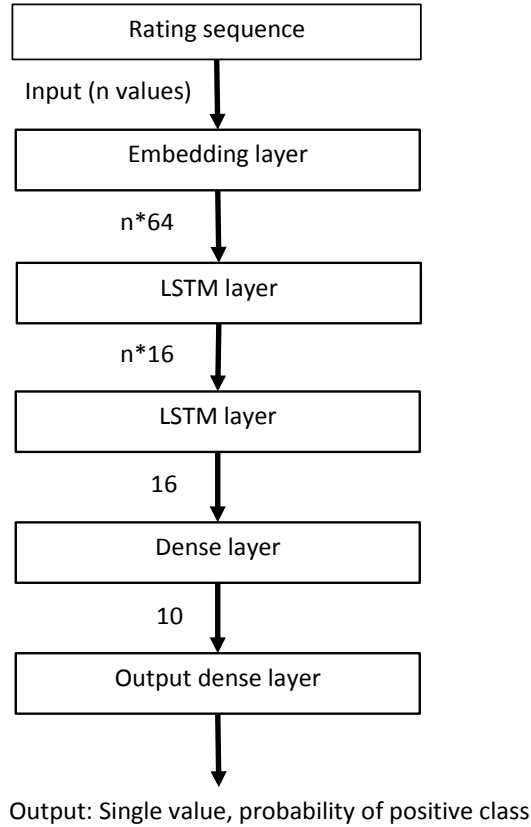
Fig. 2. Structure of proposed deep learning model.

generates a single vector representation of the ratings sequence, which is the output of the last unit of this network. This vector has 16 values. The vector representation is passed to a fully connected ANN network with 10 neurons, this layer is a classification layer whose activation function is ReLU. Its output is connected to a single neuron which constitutes the last layer of the network. This neuron has a Sigmoid activation function whose output is the probability that the input ratings sequence belongs to the positive class, that is, the contestant performance will decrease. Fig. 2 shows the network structure.

The network is trained by minimizing the cross-entropy loss function and the Adam optimizer is used to train the network.

6. Experiment and Results

To evaluate the proposed machine learning method along with the deep learning model, we conduct a real experiment. This section explains the data collection, the evaluation metrics and the detailed results.

6.1. *The Dataset*

To get data about competitive programming contestants, we use the data available from the website CodeForces.[*] This website is widely used for training people in competitive programming, moreover, it provides a rating system that allows each member to have a rate, which reflects the level of this member and changes after each round of competition conducted by the website.

Codeforces provides a public application programming interface (API)[**], this API allows access to public data from the website and its users. To collect our dataset, firstly we used the 'user.ratedList' API method to get a list of users who have participated in at least one rated contest, we are interested only in active users, so we got about 21,000 users. We could get the rating history of each user by using 'user.rating' API method, so we eliminated the users who had less than 20 ratings, leaving us with 6876 users.

We chose 10 as the length of rating sequence that can reflect the contestant performance. We generated the dataset items where each item input is a sequence of 10 ratings and the output is a binary number 0 or 1 calculated by the next 10 ratings using the function *T* as described in section 3 of this paper.

Many items may be generated by a single user, as every 20 consecutive ratings of the user can generate an item, so we used a window with a width of 20 to pass over the contestant ratings and generate the items. Eventually our dataset contained 233629 items.

The dataset is imbalanced, that is, the number of items getting better results in the future is greater than the number of contestants getting worse results. It seems this is natural as contestants try to get better results, but this fact makes our goal harder to achieve, as we are seeking contestants with worse results predicted, in order to detect them early on. Fig. 3 shows the class distribution in the dataset.

The dataset is split into two parts:

- Training set: used to train machine learning models and is 80% of the dataset.
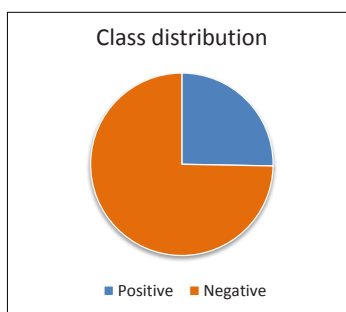- Testing set: used for evaluation and is 20% of the dataset.

Fig. 3. Class distribution in the dataset.

6.2. *Evaluation Metrics*

To evaluate the different proposed models, we are going to use well-known evaluation metrics in classification tasks.

6.2.1. *Area Under ROC Curve (AUROC)*

The receiver operating characteristics (ROC) curve is an important curve used to evaluate machine learning models. It represents the relation among sensitivity of the model with the specificity and the threshold used to choose the class. This curve reflects the performance of the model, so the area under this curve can reflect how perfect the model is.

6.2.2. *Accuracy*

The accuracy of the model is the ratio of the items that are classified correctly:

$$]accuracy = \frac{TP + TN}{N}$$

Where *TP* is the number of positive items classified correctly, *TN* is the number of negative items classified correctly, and N is the number of all classified items.

6.2.3. *Precision*

The precision of the model is the ratio of correct positive items:

$$Precision = \frac{TP}{TP + FP}$$

Where *FP* is the items that are wrongly classified as positive when they are negative.

6.2.4. *Recall*

The recall is the ratio of positive items that are correctly classified as positive:

$$Recall = \frac{TP}{TP + FN}$$

Where FN are positive items that are wrongly classified as negative. This metric is the most important one in our study, because it is more important to detect most of the contestants with performance decreasing rather than getting high classification precision.

6.2.5. *F1 Score*

*F*1 score can merge both precision and recall; it is more suitable than accuracy when data is imbalanced:

$$F1\ score = \frac{2 * Precision * recall}{Precision + recall}$$

7. Results

The six models presented in this paper are trained using the training part of the dataset, then the testing set is used to evaluate the performance of these models.

To handle the data imbalance problem of the dataset, we use class weights during the training of the models, where each class is weighted so that the weights reflect the ratio of class in the training data:

$$W_c = \frac{N}{2 * N_c}$$

Where W_c is the weight for class c, N is the total number of items in the dataset and N_c is the number of items with class c in the dataset.

Fig. 4 shows the AUROC of the models:

- Linear regression (LR).
- Random forest (RF).
- Artificial neural network (ANN).
- Multinomial Naïve Bayes (MNB).
- Gaussian Naïve Bayes (GNB).
- Probabilistic Support vector machine (PSVM).
- Deep learning model (DL).

Note that this measure cannot be calculated for support vector machines as their output is discrete, unlike other models where the output is a probabilistic continuous value that falls in range (0,1).

We can notice that the deep learning model achieved the best value for this metric. MLP and RF achieved good results as well, whereas probabilistic models had poor performance.
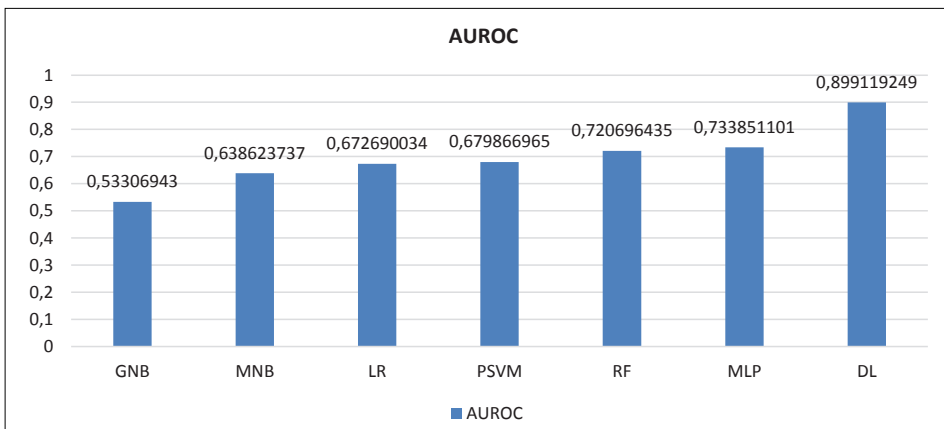
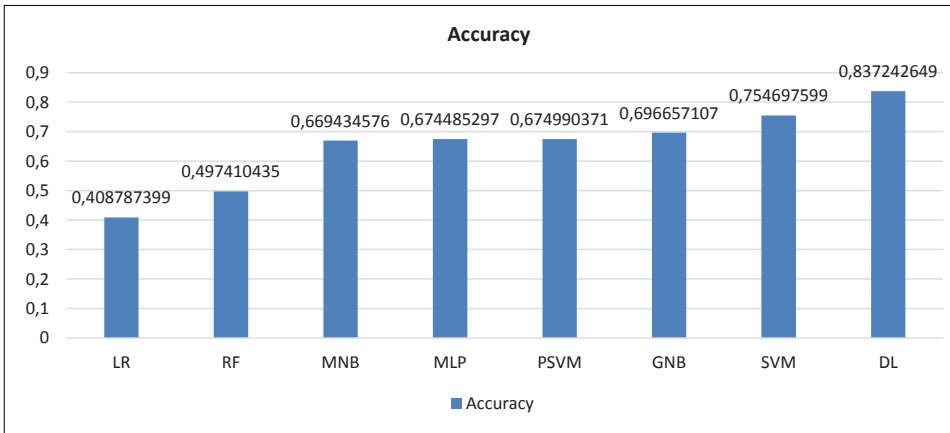Fig. 4. AUROC of different models.

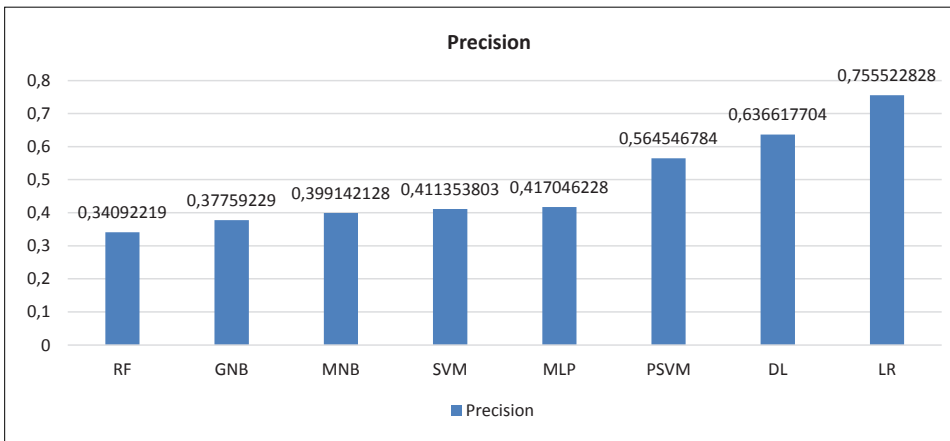Fig. 5. Accuracy of different models.

Fig. 6. Precision of different models.

Fig. 5 shows the accuracy of different models: deep learning achieved the best result for this metric, and the support vector machine had good accuracy as well, whereas linear regression had the poorest result.

Fig. 6 shows the precision of different models: linear regression outperformed all other models including the deep learning one. However, this metric does not reflect the overall performance of the model, because it is more important to detect most contestants with poor predicted performance rather than getting a high ratio of correctly detected contestants.

Fig. 7 shows the recall metric of the different models: the deep learning model has the highest recall while the random forest model also achieved good results. Recall is very important as it refers to the ratio of contestants with poor performance that are
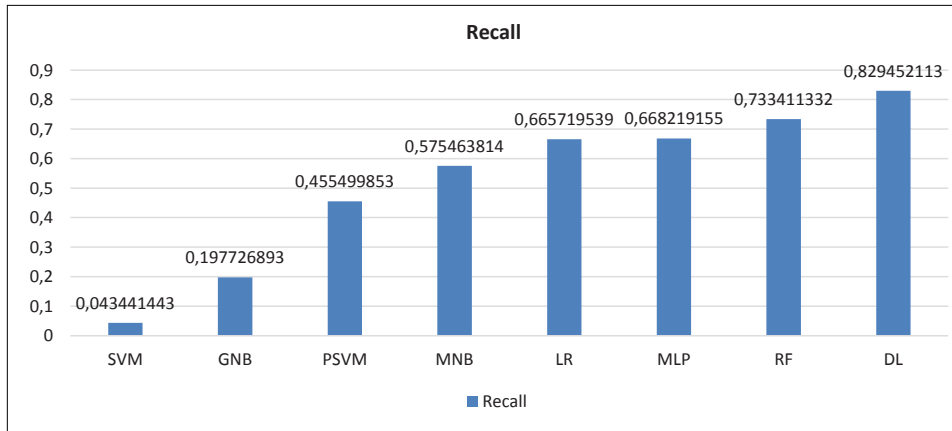
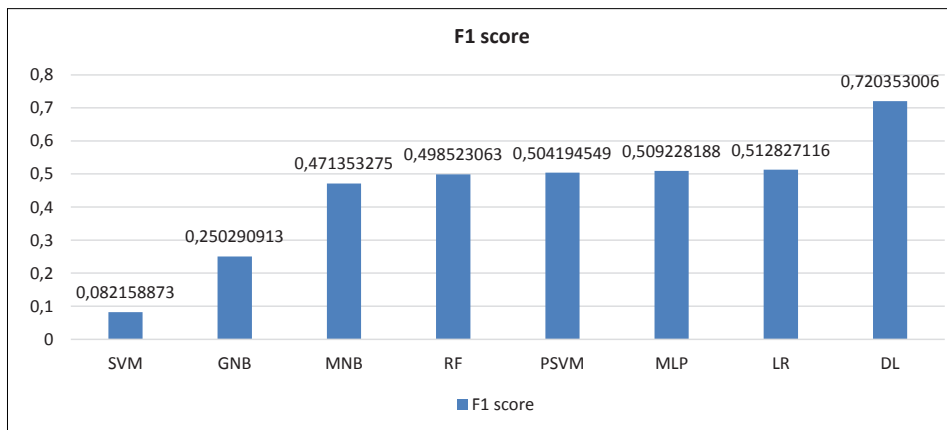Fig. 7. Recall of different models.

Fig. 8. F1 score of different models.

positively classified. The support vector machine model tends to get very few positive items, so that it has a very low recall value.

Although recall is important, precision should not be bad, so because the F1 score can reflect both precision and recall, it is more suitable than accuracy when the data is imbalanced as in our case. Fig. 8 shows the F1 score for different models. We can observe that deep learning model outperforms all other models for F1 score for a high margin.

We can see that the deep learning model achieved good results. In fact, it gets the best results for most metrics, which can be explained by the capability of the LSTM network to encode sequential data. In addition, we used a vectorized representation of ratings in this model using the embedding layer. Fig. 9 shows the ROC for this model. The height of the peak near the top left corner with high AUROC proves the efficiency of this model for this task.
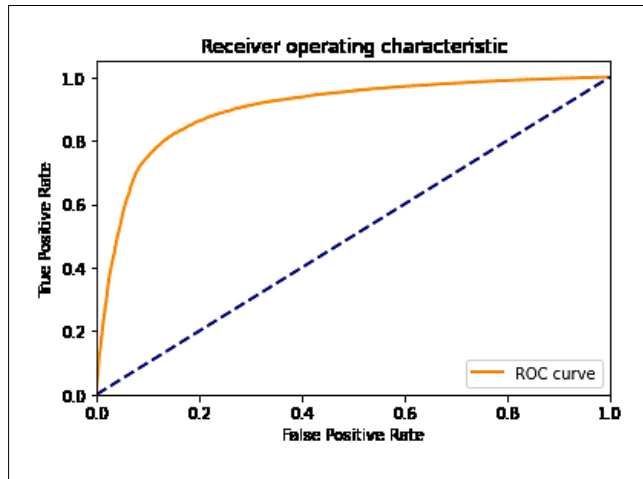
Fig. 9. ROC curve for the deep learning model.

8. Conclusion

It is useful for competitive programming coaches to track the performance of their contestants, so we can benefit through machine learning techniques to build systems that can predict the future performance of contestants and draw coaches' attention to those who are not on track to do well in future. In this paper, we presented the application of six different machine learning models in this regard; five of these models are baseline methods, and the sixth is a novel deep learning model. The results showed that it is possible to predict the future performance of contestants. Although most models achieved acceptable results, the deep learning model outperformed all models and proved to be effective and efficient.

To apply this results in reality, we have many options, but as informing contestants with the prediction of their future performance can affect their real performance; this information should be restricted to the coaches and supervisors. It can be presented as an indicator for each contestant information page, if using an information system, or it can be incorporated in any computer system used for observing contestant training process.

This study is a starting point for future development of intelligent systems that can help the process of training in competitive programming, the demographic features of contestants should be considered in future. In addition, artificial intelligence can help in many other ways in the training process like personalizing the materials or automatic content generation, which should be addressed in future research.

References

Al-Shabandar, R., Hussain, A., Laws, A., Keight, R., Lunn, J., & Radi, N. (2017). Machine learning approaches to predict learning outcomes in Massive open online courses. Paper presented at the *2017 International Joint Conference on Neural Networks (IJCNN)*.

Amra, I. A. A., & Maghari, A. Y. (2017). Students performance prediction using KNN and Naïve Bayesian. Paper presented at the *2017 8th International Conference on Information Technology (ICIT)*.

Babić, I. Đ. (2017). Machine learning methods in predicting the student academic motivation. *Croatian Operational Research Review*, 443–461.

Colchester, K., Hagras, H., Alghazzawi, D., & Aldabbagh, G. (2017). A survey of artificial intelligence techniques employed for adaptive educational systems within e-learning platforms. *Journal of Artificial Intelligence and Soft Computing Research,* 7(1), 47–64.

Combéfis, S., & Paques, A. (2015). Organising national olympiads in informatics: A review of selection processes, trainings and promotion activities. *Olympiads in Informatics,* 9.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation,* 9(8), 1735–1780.

Hussain, S., Muhsion, Z. F., Salal, Y. K., Theodoru, P., KurtoÄ lu, F., & Hazarika, G. (2019). Prediction model on student performance based on internal assessment using deep learning. *International Journal of Emerging Technologies in Learning (iJET),* 14(08), 4–22.

Jain, L. C., & Medsker, L. R. (1999). *Recurrent Neural Networks: Design and Applications*: CRC Press, Inc.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kotsiantis, S., Pierrakeas, C., & Pintelas, P. (2004). Predicting students' performance in distance learning using machine learning techniques. *Applied Artificial Intelligence,* 18(5), 411–426.

Koutina, M., & Kermanidis, K. L. (2011). Predicting postgraduate students' performance using machine learning techniques *Artificial intelligence applications and innovations* (pp. 159–168): Springer.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news,* 2(3), 18–22.

Ofori, F., Maina, E., & Gitonga, R. (2020). Using machine learning algorithms to predict students' performance and improve learning outcome: A literature based review. *Journal of Information and Technology,* 4(1), 33–55.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning,* 1(1), 81–106.

Sekeroglu, B., Dimililer, K., & Tuncal, K. (2019). Student performance prediction and classification using machine learning algorithms. Paper presented at the *Proceedings of the 2019 8th International Conference on Educational and Information Technology*.

Tan, M., & Shao, P. (2015). Prediction of student dropout in e-learning program through the use of machine learning method. *International Journal of Emerging Technologies in Learning,* 10(1).

Tanner, T., & Toivonen, H. (2010). Predicting and preventing student failure-using the k-nearest neighbour method to predict student performance in an online course environment. *International Journal of Learning Technology,* 5(4), 356.

Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., & Schmidt-Thieme, L. (2010). Recommender system for predicting student performance. *Procedia Computer Science,* 1(2), 2811–2819.

Waheed, H., Hassan, S.-U., Aljohani, N. R., Hardman, J., Alelyani, S., & Nawaz, R. (2020). Predicting academic performance of students from VLE big data using deep learning models. *Computers in Human Behavior,* 104, 106189.

Xu, J., Moon, K. H., & Van Der Schaar, M. (2017). A machine learning approach for tracking and predicting student performance in degree programs. *IEEE Journal of Selected Topics in Signal Processing,* 11(5), 742–753.

A. Alnahhas is a teacher at the faculty of information technology engineering, Damascus University, he holds a Ph.D. in artificial intelligence, he was involved in the training and preparation of national Olympiad in informatics since 2005, he was involved in many competitive programming activities, he has been the leader of Syrian delegation to IOI for many years.

N. Mourtada is an information technology engineering student at the department of artificial intelligence, faculty of information technology engineering, Damascus University.

Latest Algorithms on Particular Graph Classes

Phan Thuan DO[1,*], Ba Thai PHAM[1,*], Viet Cuong THAN[2]
[1]*Hanoi University of Science and Technology*
[2]*University of Nebraska-Lincoln*
e-mail: thuandp@soict.hust.edu.vn, thai9cdb@gmail.com, cthan2@huskers.unl.edu

Abstract. Many optimization problems such as Maximum Independent Set, Maximum Clique, Minimum Clique Cover and Maximum Induced Matching are $\mathbb{NP}$-hard on general graphs. However, they could be solved in polynomial time when restricted to some particular graph classes such as comparability and co-comparability graph classes. In this paper, we summarize the latest algorithms solving some classical NP-hard problems on some graph classes over the years. Moreover, we apply the α-redundant technique to obtain linear time $\mathcal{O}(|V|)$ algorithms which find a Maximum Induced Matching on interval and circular-arc graphs. Inspired of these results, we have proposed some competitive programming problems for some programming contests in Vietnam in recent years.

Keywords: graph classes, interval graph, circular-arc graph, co-comparability, maximum induced matching.

1. Introduction

Despite the fact that the study field of particular graph classes has been skyrocketing over the years, there is still modest number of competitive programming problems inspired by this field for Olympic programming contests, especially for high school contests. One reason is that the particular graph classes are excluded in the IOI syllabus. However, many particular properties of these graph classes can be expressed as some other knowledge included in the syllabus. We have explored some of these properties to propose problems for some programming contests in Vietnam in recent years. These problems are not mentioned about particular graph classes by being stated as practical situations. They can be consequently solved by knowledge included in the IOI syllabus.

Usually, $\mathbb{NP}$-hard problems may become much easier when restricted on particular graph classes. Exploring this way, we first study graph optimization problems such as Maximum Independent Set, Maximum Matching, Maximum Clique, Minimum Clique

---

* Corresponding authors

Cover. We summarize latest algorithms for these problems on some co-comparability graph classes such as interval, permutation and trapezoid graphs.

Besides, in the literature, there exist only a few trivial algorithms of finding Maximum Induced Matching (MIM) on particular graph classes. This problem was first proposed in 1989 by Cameron (Cameron, 1989). While the maximum matching problem can be solved in polynomial time in an arbitrary graph, the MIM problem is a $\mathbb{NP}$-Hard problem, even for bipartite graphs. The MIM problem can be trivially solved in polynomial time for interval graphs and chordal graphs (Cameron, 1989), circular-arc graphs (Golumbic, 1993), Trapezoid graphs and co-comparability graphs (Golumbic and Lewenstein, 2000), Asteroidal-triple-free graphs (Cameron, 2004; Chang, 2001), Weakly chordal graphs (Cameron *et al.*, 2003), Interval-filament graphs (Cameron, 2004). There are algorithms that can find MIM in linear time $\mathcal{O}(|E| + |V|)$ in chordal graphs (Brandstädt and Hoàng, 2008), interval graphs (Golumbic and Lewenstein, 2000), tree graphs (Golumbic and Lewenstein, 2000) and permutation bipartite graphs (Chang, 2001). In this paper, we present two $\mathcal{O}(|V|)$ algorithm solving the MIM problem on interval and circular-arc graphs. Our approach is to use the α-redundant technique to reduce the search space while still preserving optimal solutions.

In the last section, we describe some of our competitive programming problems proposed for Vietnam Team Selection Tests (TST) in recent years. These problems are inspired by the latest results on circular-arc, trapezoid and disc graphs for the problems of maximum induced matching and minimum vertex cover.

2. Preliminary

2.1. $\mathbb{NP}$-*Hard Graph Problems*

Given a graph $G = (V, E)$, a set $S \subseteq V$ is called an independent set of G if no two members of S are adjacent. The number of vertices in a maximum independent set (MIS) of G is called the independence number, denoted by $\alpha(G)$.
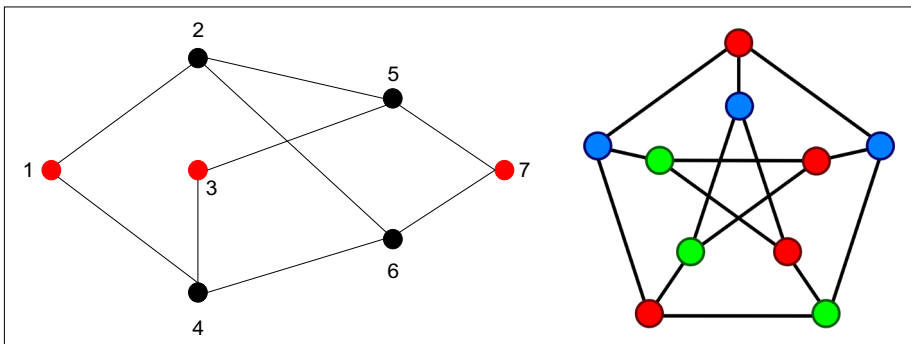

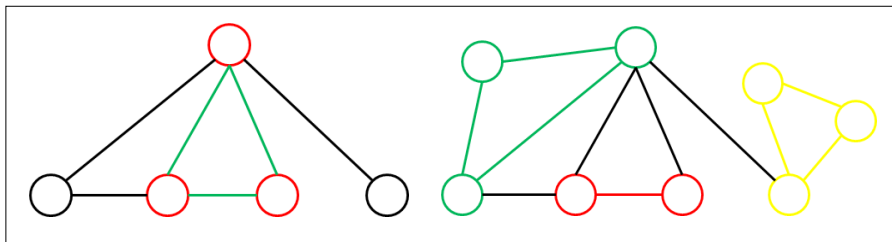Fig. 1. An illustration of Independent Set and Graph Coloring.

Fig. 2. An illustration of Clique and Clique Cover.

A subset C of V is a clique if and only if every two vertices of C are adjacent. A clique of graph G corresponds to an independent set of the complement graph $\overline{G}$. The cardinality of a maximum cardinality clique (MC) is called $\omega(G)$. A clique cover of G is a partition of the set V into cliques. The number of cliques in a minimum clique (MCC) cover of G is denoted by $k(G)$.

A coloring of a graph is an assignment of labels, also called color, to each vertex such that no two adjacent vertices share the same color. The minimum number of colors needed to assign a graph subject the constraint is called the chromatic number of that graph and is denoted by $\chi(G)$. Vertices with the same color of G are in a clique of the complement graph $\overline{G}$ of G. Hence a coloring of G is a minimum clique cover of $\overline{G}$.

These four problems MIS, MC, MCC, and Coloring have been known to be $\mathbb{NP}-$ hard in general graphs. However, many of them can be solved efficiently with polynomial time complexity in following particular graph classes.

2.2. *Particular Graph Classes*

A graph G is comparability if there is a transitive orientation, an assignment of directions to the edges of the graph, *i.e.* an orientation of the graph, such that the adjacency relation of the resulting directed graph is transitive: whenever there exist directed edges (x, y) and (y, z), there must exist an edge (x, z). A co-comparability graph is a complement of the comparability graph. The MC and Coloring problem in comparability graphs can be solved in linear time $\mathcal{O}(|E| + |V|)$ using the lexicographic depth-first search algorithm (Golumbic, 2004) while a maximum independent set and minimum clique cover could be found by using maximum flow algorithms (Golumbic, 2004). These algorithms could be taken advantage of solving MIS, MC, MCC and Coloring problems in co-comparability graphs.

A graph is an intersection graph if each vertex corresponds to a set and two vertices are adjacent iff their set share same members. If each set is an interval on a line, the graph is called an interval graph.

A simple greedy $\mathcal{O}(|V|)$ algorithm can be used to solve the MIS and MCC problem in interval graphs, with an assumption that every interval in the input is sorted by their left ends. The Coloring and MC problem can be solved in $\mathcal{O}(|V| \log |V|)$.
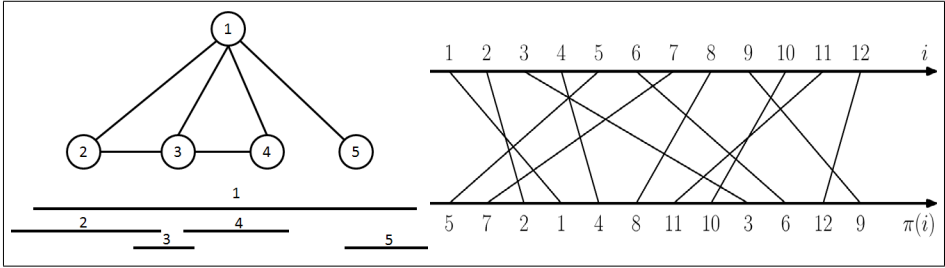
Fig. 3. An illustration of Interval Graph and Permutation Graph.

A permutation graph is a graph whose vertices represent the elements of a permutation, and whose edges represent pairs of elements that are reversed by the permutation. Permutation graphs may also be defined geometrically, as the intersection graphs of line segments whose endpoints lie on two parallel lines.

Permutation graphs are both comparability and co-comparability. There is an $\mathcal{O}(|V| \log \log |V|)$ algorithm based on the longest increasing subsequence to solve the MIS problem in permutation graphs. This algorithm can be used to find MC, MCC and Coloring also in $\mathcal{O}(|V| \log \log |V|)$.

A trapezoid graph is an intersection graph of trapezoids in which two sides line on two parallel lines. The MIS and MCC problem in this graph class can be solved in $\mathcal{O}(|V| \log \log |V|)$ using the sweep line technique (Felsner *et al.*, 1997). A MC and Coloring could be found in $\mathcal{O}(|V| \log |V|)$.
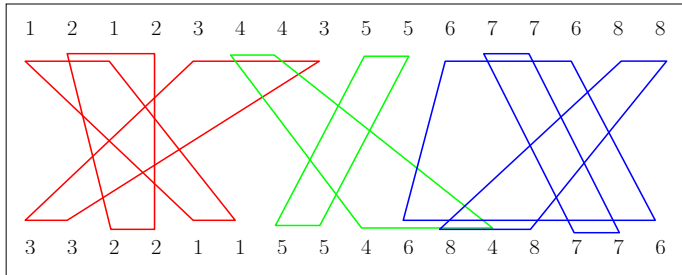
Fig. 4. An illustration of Trapezoid Graph.

Table 1
Latest algorithms on some particular graph classes

Problem	Comparability	Co-comparability	Interval	Permutation	Trapezoid
MC	$\mathcal{O}(|E| + |V|)$	Polynomial	$\mathcal{O}(|V| \log |V|)$	$\mathcal{O}(|V| \log \log |V|)$	$\mathcal{O}(|V| \log |V|)$
Coloring	$\mathcal{O}(|E| + |V|)$	Polynomial	$\mathcal{O}(|V| \log |V|)$	$\mathcal{O}(|V| \log \log |V|)$	$\mathcal{O}(|V| \log |V|)$
MIS	Polynomial	$\mathcal{O}(|E| + |V|)$	$\mathcal{O}(|V|)$	$\mathcal{O}(|V| \log \log |V|)$	$\mathcal{O}(|V| \log \log |V|)$
MCC	Polynomial	$\mathcal{O}(|E| + |V|)$	$\mathcal{O}(|V|)$	$\mathcal{O}(|V| \log \log |V|)$	$\mathcal{O}(|V| \log \log |V|)$

3. The α - Redundant Method

Given a graph $G = (V, E)$, the k-th power graph of G is denoted by $G^k = (V, E')$ having the same vertex set with G. Two vertices u, v in G^k are adjacent iff there exists a path from u to v of length less than or equal to k. Let $L(G)$ denote the line graph of G, *i.e.*, each edge of G is a vertex of $L(G)$, two vertices of $L(G)$ are adjacent iff two corresponding edges share a common endpoint. An edge set $M \subseteq E$ is called a matching of G iff there does not exist a pair of edges in M with a common vertex. An induced matching of G is a matching where the distance between two arbitrary vertices in two different edges is at least two.

An induced matching of a graph G corresponds with an independent set of $L(G)^2$. So there will be a polynomial complexity algorithm for MIM whenever MIS of a graph can be found in polynomial time. In some circumstances, avoiding fully constructing the graph $L(G)^2$ may lead to better time complexity.

A vertex of G is α - redundant iff its removal does not affect the size of the MIS in G. The approach is to remove α - redundant vertices from $L^2(G)$ before finding a MIS of the remaining graph L^*.

4. Maximum Induced Matching in Interval Graphs

In this section, we propose a linear time $\mathcal{O}(|V|)$ algorithm solving the MIM problem in interval graphs based on the α-redundant technique with an assumption that every interval in the input is sorted by their left ends. The algorithm will be improved to find a MIM in circular-arc graphs with the same computational complexity $\mathcal{O}(|V|)$. For any interval u, we denote the coordinate of its left and right end by $l(u)$ and $r(u)$, respectively. We first give an important property.

Lemma 1. *If G is an interval graph, the graph $L^2(G)$ is also an interval graph.*

Proof. For each edge (u, v), with $l(u) \leq l(v)$, we draw a line from $l(u)$ to $\max\{r(u), r(v)\}$, the right end of the union line of u and v. $L^2(G)$ is an interval graph in which
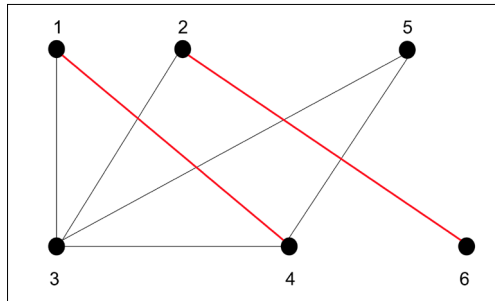

Fig. 5. An illustration of induced matching.

each vertex of $L^2(G)$ corresponding with its union line. If (u, v) and (w, z) are adjacent, there is at least one edge between 2 two pairs of vertices (u, v) and (w, z). Assume that the interval u and w cut each other. So the union interval (u, v) and (w, z) also intersect each other. It is trivial that if two union interval (u, v) and (w, z) are not adjacent, there does not exist any edge connect u or v to one of $\{w, z\}$.

4.1. *The Construction of* L^*

From the aforementioned lemma, the graph $L^2(G)$ is an interval graph so the new graph L^* is an interval graph as well. Following the same idea in finding a MIM in permutation graphs, we will remove α - redundant vertices from $L^2(G)$. For each interval u, the algorithm will find the *optimal* interval v of u. The definition of the optimal interval can be express as follows:

Definition 1. Given an interval u, the optimal interval v of u is interval that $r(u) \geq l(v) \geq l(u)$ and union interval (u, v) is shortest.

If there exists an optimal interval v of interval u, every union interval of u and an interval v 'smaller' than u except (u, v) is α - redundant. This property can be proved by the argument exchange method because every interval in the maximum independent set of $L^2(G)$ can be replaced by a union interval of a vertex and its optimal interval. So the number of vertices in L^* is at most n and finding the maximum independent set of an interval graph costs $\mathcal{O}(|V|)$ if the intervals are sorted by their left ends.

4.2. *Algorithm*

The algorithms finding a MIM of an interval graph can be divided into two steps, the first one is constructing the graph L^* and the second is finding a MIS of L^*. The intervals in the input are already sorted by their left ends.

Algorithm 1 $Optimal(V)$

  1: **Input:** Set of intervals V.
  2: **Output:** Optimal interval of each members in V
  3: $optimal\_vertex(u) \leftarrow null$ for every $u \in V$
  4: Stack $S \; \emptyset$
  5: **for** $u \in V$ **do**
  6:     **while** $S = \emptyset$ **do**
  7:         $v \leftarrow S.top$
  8:         **if** $r(v) < r(u)$ **then**
  9:             Break
10:         **end if**
11:         $optimal\_vertex(v) \leftarrow u$
12:         $S.pop$

13:     **end while**
14:     **if** $S \neq \emptyset$ and $l(u) \leq r(S.top)$ **then**
15:         $optimal\_vertex(S.top) \leftarrow u$
16:     **end if**
17:     $S.push(u)$
18: **end for**
19: Return $optimal\_vertex$

Algorithm 2 *MIM Interval*(V)

 1: **Input:** Set of intervals V
 2: **Output:** An MIM of the interval graph $G(V)$
 3: $optimal\_vertex \leftarrow Optimal(V)$
 4: $V^* \leftarrow \emptyset$
 5: **for** $u \in V$ **do**
 6:      **if** $optimal\_vertex(u) \neq null$ **then**
 7:          $v \leftarrow optimal\_vertex(u)$
 8:          Add the interval $u \cup v$
 9:      **end if**
10: **end for**
11: Stack $S \leftarrow \emptyset$
12: **for** $u \in V^*$ **do**
13:      **if** $S = \emptyset$ **then**
14:          $v \leftarrow S.top$
15:          **if** v does not cut u **then**
16:              $S.push(v)$
17:          **end if**
18:      **end if**
19: **end for**
20: Return S

4.3. *Complexity*

Each interval in V will be pushed and popped at most once. So, the time complexity of the algorithm is $\mathcal{O}(|V|)$

5. MIM in Circular-arc Graphs

The algorithm finding a MIM in interval graphs can be extended to solve the MIM problem in circular-arc graphs. Like in interval graphs, we give first the important property of $L^2(G)$ of a circular-arc graph G.

Lemma 2. *The graph* $L^2(G)$ *of a circular-arc graph* G *is also a circular-arc graph.*

This lemma can be proven using the same technique as in interval graphs. Assume that there is an origin in the circle, every end of the arc are coordinated. l is the left end of an arc (l, r) if and only if the arc from l to r is clockwise. We propose an $\mathcal{O}(|V|)$ algorithm to find a MIM of a circular-arc graph with the assumption that the arcs are already sorted by their left end and their length. The idea of the algorithm is always to find an optimal neighbor corresponding with each vertex. We first re-coordinate the arcs to identify the stating arc on the circular, and then applying the same algorithm for interval graphs from the starting arc.

Algorithm 3 $Optimal(V)$

1: **Input:** V is a sorted set of arcs.
2: **Output:** Optimal neighbor with each vertex of V.
3: $u0$ is the shortest arc of V
4: Re-coordinate arc in V with the origin is the left end of u_0.
5: Stack $S \leftarrow \emptyset$
6: $optimal\ vertex(u) \leftarrow null$ for every $u \in V$
7: **for** $u \in V$ **do**
8: 　　**while** $S \neq \emptyset$ **do**
9: 　　　　$v \leftarrow S.top$
10: 　　　　**if** $r(v) < r(u)$ **then**
11: 　　　　　　Break
12: 　　　　**end if**
13: 　　　　$optimal\ vertex(v) \leftarrow u$
14: 　　　　$S.\text{pop}$
15: 　　**end while**
16: 　　**if** $S \neq \emptyset$ and $l(u) \leq r(S.top)$ **then**
17: 　　　　$optimal\ vertex(S.top) \leftarrow u$
18: 　　**end if**
19: 　　$S.\text{push}(u)$
20: **end for**
21: Update $optimal\ vertex$ of arcs intersecting u_0
22: Return $optimal\ vertex$

The correctness of the algorithm is proven by the following lemma.

Lemma 3. *The procedure* $Optimal(V)$ *return the optimal arc of each arc in* V.

Proof. With an arc u, if $r(u) \geq l(u)$, the work finding its optimal neighbor is similar to interval graph. Consider the arc u with $r(u) < l(u)$, which means this arc contains the origin, we will prove that the optimal neighbor of u can only be an arc v that $r(v) < l(v)$, or be u_0. If the optimal neighbor of u is an arc w that $r(w) > r(u)$ and $r(w) > l(w)$, which mean this arc does not contain the origin, the length of the union arc (u, w) is smaller than that of (u, u_0). So, $r(w) < r(u_0)$ and $l(w) < l(u_0)$. This implies length of w is smaller than length of u_0, which is contrary to our choice that u_0 is the shortest arc.

6. Some Induced Competitive Programming Problems

6.1. *Circular-arc (Vietnam TST 2018)*

Given a circle defined by the center coordinates (x, y), radius r and n lines, in which the ith line is determined by the equation $a_i x + b_i y + c_i = 0$. There is no secant passing through the center of the circle. A secant if cut the circle at 2 points A and B, the arc AB small of the circle is called the characteristic arc of that line. Note that if the secant touches the circle, the characteristic arc degrades to 1 point. Next, to examine the relationship between arcs, Hai constructs a simple undirected graph $G = (V, E)$, where each vertex of V corresponds to a typical arc of the circle, and the set edge E consists of all the edges connecting the two vertices in V where the two characteristic arcs correspond to them. We call the path length d between two edges e and f as a sequence of edges $(e, g_1, g_2, ..., g_d, f)$ so that two consecutive edges in this sequence have a common vertex. The distance between the two sides e and f is the length of the shortest path between them. If there is no path between e and f, the distance between them is set to $+\inf$. Hai wants to find the set of edges $E' \subseteq E$ with the largest cardinality such that the distance between any two edges in E' is at least 2. See Fig. 6 for an example.

The problem can be directly solved by the $\mathcal{O}(n)$ MIM algorithm on Section 5 plus the time $\mathcal{O}(n \log n)$ to sort arcs by their left end and their length. Hence the size n of input can go to 10^6.
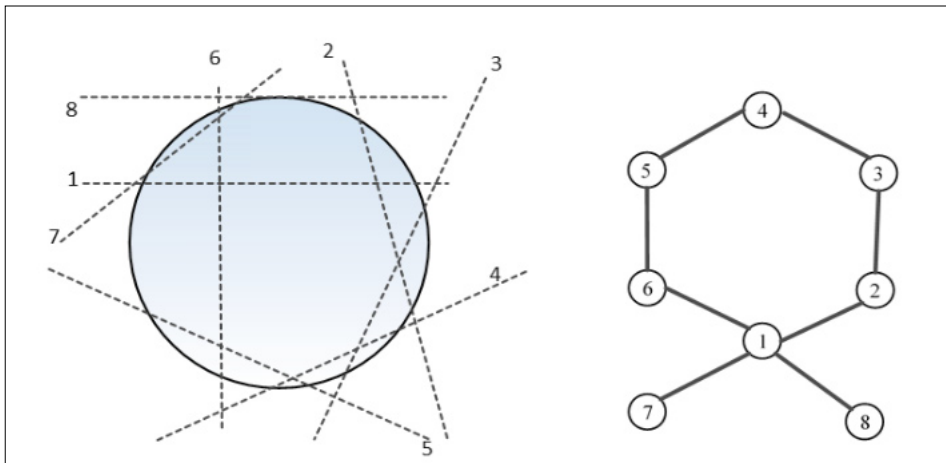

Fig. 6. An example of Problem 6.1.

6.2. *ESEA (Vietnam TST Camp 2018)*

The eastern territorial sea has n critical area ($n \leq 10^6$). The entire territorial sea is depicted on a map of coordinates, where each critical area is represented by a rectangle with vertices at points with integer coordinates. In preparation for the unprecedented training session "ESEA" at sea, the Naval Military Command is planning a battle on the original map of simulated territorial waters. n pair of detectors, each pair of detectors (δ_1, δ_2) at two critical points:

- δ_1 detector set at coordinates (x_1, y_1) is capable of detecting objects within its left lower quadrant, *i.e.* all points with coordinates (u, v) satisfies: $u \leq x_1$ and $v \leq y_1$.
- δ_2 detector set at coordinates (x_2, y_2) is capable of detecting objects within its right higher quadrant, i.e. all points with coordinates (u, v) satisfies: $u \geq x_2$ and $v \geq y_2$. Know that $x_1 \leq x_2$, $y_1 \leq y_2$.

Two pairs of detectors i and j are called **interconnected** if both detectors of j pair are fully within the detection range of either detector of i.

The military command requires the collection of sets of detectors into at least groups so that each pair must belong to exactly one group and in each group, there are no two pairs that are interconnected. See Fig. 7 for an example.

Hint. Consider two parallel lines X and Y. Each point (x_0, y_0) corresponds with a line connecting the point x_0 on X with y_0 on Y. Each detector (δ_1, δ_2) forms a trapezoid. See Fig. 8 or an Illustration. Two detectors are interconnected iff two trapezoids are separate. The problem becomes to find the MCC of the trapezoid graph constructed from the trapezoid model corresponding to the detectors.
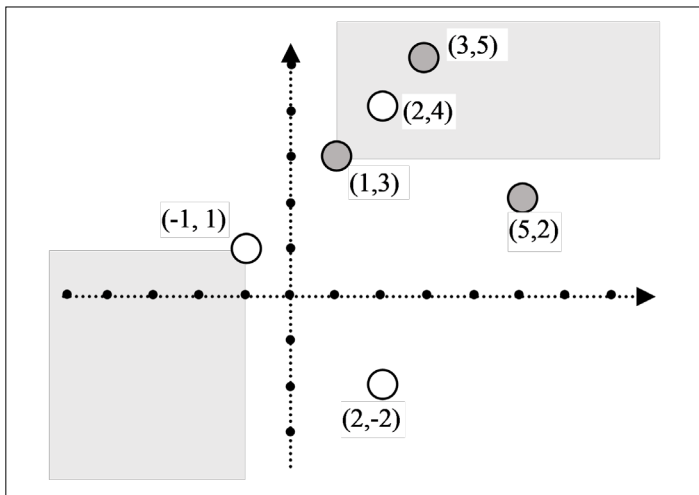

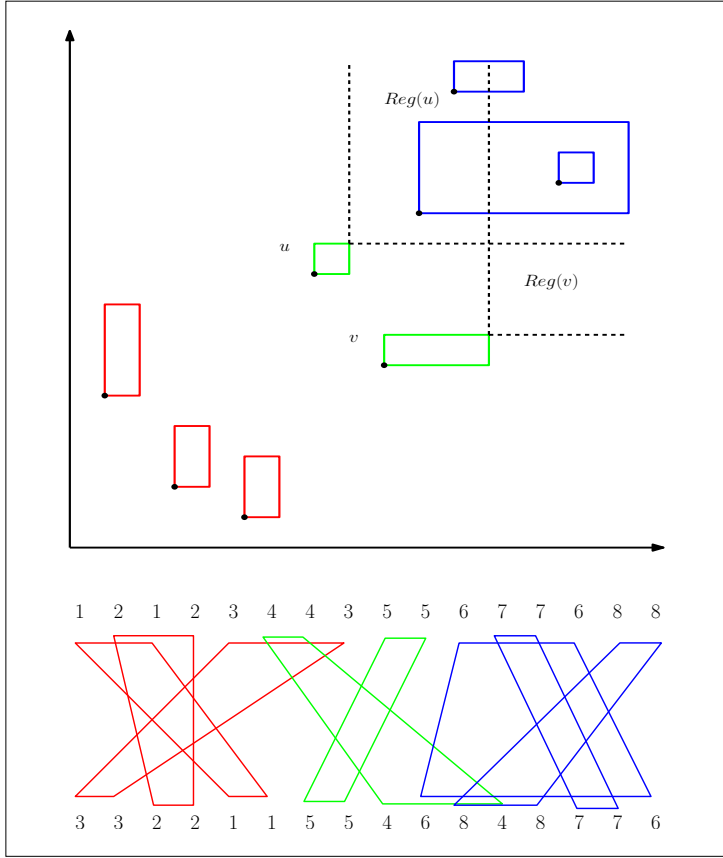Fig. 7. An example Problem 6.2.

Fig. 8. A Trapezoid Diagram.

We first sort the trapezoids by their left most point. Denoted the cardinality of the maximum independent set of the trapezoid graph $1, 2, \ldots, i$ containing i by f_i. Let $F = \max_{i=1}^{n} f_i$, then the partition $S_k = \{i \in \{1, 2, \ldots, n\} : f_i = k\}$, $k = 1, 2, \ldots, F$ is the MCC. Consider two trapezoid i, j such that $i < j$, if i and j are not adjacent then $f_j \geq f_i + 1$, so i and j are in 2 different subset of S. Therefore S is a clique cover.

Let P be an arbitrary clique cover and IS be an arbitrary independent set. Since any two members of IS must be in 2 different clique then $|P| \geq |IS|$. Otherwise there exists an independent set with size f_i for all $i = 1, 2, \ldots, n$, then $|P| \geq \max_{i=1}^{n} f_i$, in other words $|P| \geq |S|$.

We have a recurrence equation: $f_i = \max_{j<i}^{i \cap j = \emptyset} f_j + 1$. Using the same technique as in finding a longest increasing subsequence, we obtain the computational complexity $\mathcal{O}(n \log n)$ with Binary Indexed Tree.

6.3. *The Battle on the River (Vietnam TST 2019)*

Hung is simulating a battle on the river as follows. The map of the river is shown on the coordinate plane. The two banks of the river are given by two parallel lines $x = a$ and $x = b$. There are n piles (numbered from 1 to n) is nailed on the river section, pile i is nailed at the point of integer coordinates (x_i, y_i). Let c and d be the largest and the smallest ordinate respectively. To simplify the problem, each boat battle is considered a circle of diameter R. Thus, a boat when entering between two piles $A = (x_A, y_A)$, $B = (x_B, y_B)$ will be stuck if its diameter is larger than the distance between points A and B. A boat can cross the river section if it finds a way to move from one point of the river with the ordinate $c + R$ passing through the piles without stuck to reach any point with ordinate $d - R$. Finding the largest value of R so that Hung can cross the river. Constraint $n \leq 10^5$.

Hint. Consider each pile is a circle with radius r. We can construct an intersection graph $G(r)$ with each vertex corresponding to a circle or a bank. Our problem can be reduced to finding the largest value of r such that two banks are not in the same connected component.

6.4. *Building (VOI 2020)*

There are n buildings in Alice's city. In the Cartesian coordinate, a building is represented by a rectangle with sides parallel to the coordinate axes. Two buildings are adjacent if the intersection of their sides is not empty. There is a short path between each
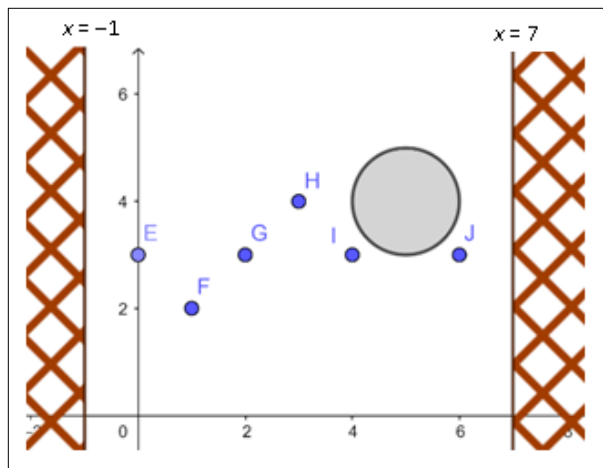

Fig. 9. An example of Problem 6.3.

pair of adjacent buildings. Alice really likes the architectures of the buildings in the city and she usually walks along those short paths. After a few days, she relies some paths are unique-paths. A path e between two building u and v is unique-path if after going from u to v, there is no way to come back to u without walking through e again. For each pair of buildings (u, v) which is a unique-path, Alice calculates the maximum number of buildings she can visit for u and for v with the assumption that the path (u, v) is closed, we call those numbers d_u and d_v respectively.

Given the coordinate of each rectangle, help Alice find the pair (u, v) which has an unidirectional path between them and the absolute difference of d_u and d_v is minimum. See Fig.10 for an example.

Hint. The problem is related to interval graphs.

1. Construct the graph G representing the adjacent relation between buildings:
    (a) Sort all rectangles in a list by the ascent order of x - coordinate;
    (b) Sort all rectangles in another list by the ascent order of y - coordinate;
    (c) For each rectangle, find its adjacent list by the two sorted lists above. The number of edges in G is only a linear function of the number of rectangles.
4. Use Tarjan's algorithm to find all bridges.
5. Find the bridge (u, v) with minimum absolute difference between d_u and d_v. The time complexity is $\mathcal{O}(n \log n)$.
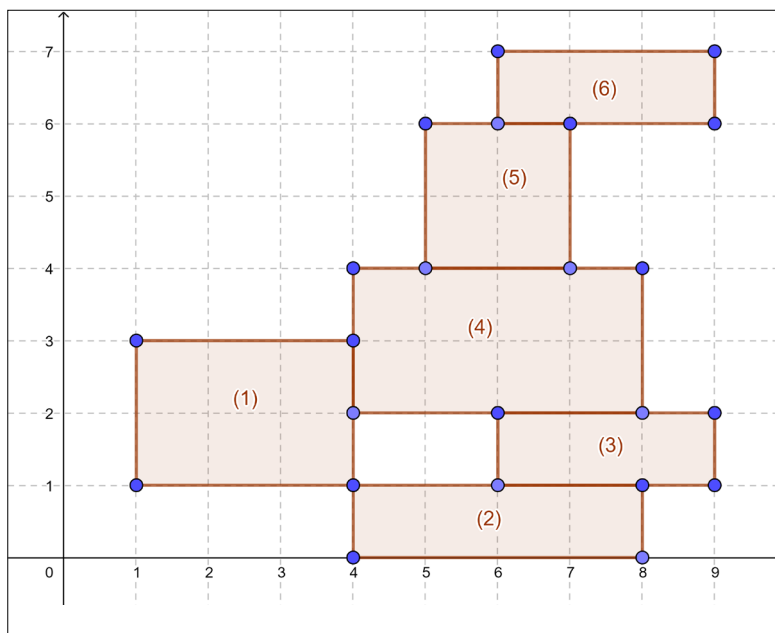

Fig. 10. An example of Problem 6.4.

7. Conclusion

Although these induced problems are related to particular graph classes that are excluded in the IOI syllabus, our proposed algorithms to solve these problems are in the scope of the syllabus.

Throughout the paper, our approach of *removing the α-redundant vertices* is proved to be effective to reduce the complexity of the algorithms for some problems on particular graph classes. In the future, we intend to apply this method for other suitable problems on some particular cases of graph theory.

Acknowledgment

References

Brandstädt, A. and Hoàng, C. T. (2008). Maximum induced matchings for chordal graphs in linear time. *Algorithmica*, 52(4), 440–447.

Cameron, K. (1989). Induced matchings. *Discrete Applied Mathematics*, 24, 97–102.

Cameron, K. (2004). Induced matchings in intersection graphs. *Discrete Mathematics*, 278, 1–9.

Cameron, K., Sritharanb, R., and Tangb, Y. (2003). Finding a maximum induced matching in weakly chordal graphs. *Discrete Mathematics*, 266, 133–142.

Chang, J. M. (2001). Induced matchings in asteroidal triple free graphs. *Discrete Applied Mathematics*, 132, 67–78.

Felsner, S., Muller, R., and Wernisch, L. (1997). Trapezoid graphs and generalizations, geometry and algorithms. *Cornell Family Papers*.

Golumbic, M. C. (1993). Irredundancy in circular arc graphs. *Discrete Applied Mathematics 4*, pages 79–89.

Golumbic, M. C. (2004). *Algorithmic Graph Theory and Perfect Graphs*, Volume 57. Elsevier.

Golumbic, M. C. and Lewenstein, M. (2000). New results on induced matchings. *Discrete Applied Mathematics*, 101, 157–165.

P.T. Do is currently Associate Professor and Deputy-Head of Department of Computer Science at Hanoi University of Science and Technology. He is also a member of the national committee for selecting, training and leading Vietnamese IOI/APIO/ICPC Teams. His current research interests include combinatorics, theory of graphs and applied algorithms in various practical problems such as logistics, network, artificial intelligence and bioinformatics.

B.T. Pham is currently a last year student of the talented engineer program at Hanoi University of Science and Technology. He participated in many *ICPC Asia Pacific Regional Contests* during his 5 academic years. He had some publications about graph theory.

V.C. Than is currently a Master student at University of Nebraska-Lincoln. He participated in *ICPC Asia Pacific Regional Contests* and in *ICPC North Central North American Regional Contest*. He had some publications about graph theory and game theory.

The Technology of Differentiated Instruction in Text Programming in Elementary School Based on the Website dl.gsu.by

Michael DOLINSKY, Mariya DOLINSKAYA
Faculty of Mathematics and Technologies of Programming, F. Skorina
Gomel State University, Sovetskaya str., 104, Gomel. 246019. Republic of Belarus
e-mail: dolinsky@gsu.by, mkugejko@gsu.by

Abstract. The technology of teaching text-based programming on the basis of the website DL.GSU.BY is described. The main advantages of the technology include: "zero entry threshold", training adapted to the pupil, many years of practical experience, effectiveness and scalability.

Keywords: distance learning, textual programming, primary school, Olympiad in Informatics, website DL.GSU.BY, F. Skorina Gomel State University.

Introduction

The authors have been actively preparing schoolchildren and students for Olympiads in Informatics and programming for many years (Dolinsky, 2016). Since 1997, this work has been carried out on the basis of the computer science cabinet of secondary school No. 27 in Gomel. Since 1999, this work has been actively supported by the distance learning website DL.GSU.BY of F. Skorina Gomel State University. An important distinguishing feature of the authors' approach is the built-in text-based training system in elementary school, the description of which this work is devoted to, emphasizing the following advantages of the authors' approach:

- Zero entry thresholds.
- Propaedeutics of text programming.
- Developing interesting differentiated training.
- Task – oriented training.
- Minimalist approach to theory.
- Regional programming Olympiads for pupils in grades 1–4.
- Preparation for the Olympiads of grades 5–8 (general tasks with the Olympiads of grades 5–8).

- Problems in school mathematics, Olympiad mathematics and informatics mathematics.
- Competitions motivating to permanent studies.
- Many years of practical experience.
- Experience-based training scalability.
- Productivity.
- Low requirements for teacher professional qualifications.
- "Accelerated Course 2013".
- Support for the transition to the study of C ++.

Zero Entry Thresholds

A programming training system based on the DL.GSU.BY website was conceived as a means of helping a teacher teaching in a computer class. But since in the learning process we permanently reduced the age at which classes began (grade 10, grade 8, grade 5, grade 1), it became fundamentally important to ensure selfeducation, because otherwise the effectiveness of joint learning of a group of children would drop sharply. Moreover, in the secondary school No. 27 of the city of Gomel, the work is done frontally – that is, with all elementary school pupils whose parents stated in writing that they want informatics classes to be held with their children. As a rule, almost 100% of parents write such statements.

In other words, in the secondary school No. 27 of the city of Gomel we teach all pupils, not selected ones. Children who come to the first grade of our school are differently prepared for learning. This forced us to develop the educational system in such a way that not only well-prepared children, but also everyone else, would participate in our classes with interest and learn effectively. All first graders begin their studies with the course "Informatics 2015" (2015 means that it was formed and fixed in 2015) in the package of tasks "Learning to think 2012" (Dolinsky, 2014). This package is aimed at developing thinking skills, which helps to learn more productively in the future. The tasks of this package with five levels of difficulty develop the following basic mental operations:

- **Operations on pairs:** comparison, ordering, association.
- **Operations on sets:** union, intersection, subtraction.
- **Operations on the set:** classification, structuring, generalization.
- **Logical operations:** negation, conjunction, disjunction, equivalence, implication.
- **Complex operations:** synthesis, memorization, analysis, imagination, analogy, abstraction, positioning.

All tasks are focused on the ability to be performed by children who cannot read, for which they are presented in the form of a picture, some of the components of which need to be moved to another location with the mouse or just click on them.

If the pupil cannot do the task, there is a button "I don't know", which transfers him to the tree of facilitating tasks. You can return back by completing the assignments, or by clicking the "I understand" button. Note that the "Learning to Think 2012" task package contains 620 main tasks and 1408 along with facilitating tasks.

In addition, for children who have systematic problems with some kind of mental operation, there are special task packages "Technical minimum. Differences", "Technical minimum. Analogy", etc., which are designed to improve the performing skills of such tasks from level zero to the ability to complete tasks in the main course "Learning to think".

For children who have systematic problems with all tasks, there is a special introductory package of tasks "Technical minimum. Learning to think – 0".

For children who do not own a mouse, there is a special package of tasks "Technical minimum. Learning to work with the mouse".

Finally, for children who needed a lot of help with the Learning to Think 2012 course, we created the Learning to Think (Quickly) course, which consists of 128 key tasks of the "Learning to Think" course (544 tasks with facilitators). Repetition is the mother of learning!

Thus, we have ensured that every first-grader of the secondary school No. 27 goes to computer science classes with pleasure, and at the same time, everyone is moving towards the development of basic mental skills to the level that provides effective follow-up training. Actually, in our system, children who have learned to walk and talk, that is, from 3–4 years old, can be engaged. And practice has shown that such classes can be conducted at home alone or with minimal help from parents.

Propaedeutics of Text Programming

Historically, text programming classes start with Pascal. The first barriers that must be overcome at the beginning of training are:

- Remember the order of keywords in the program (program, var, longint, begin, readln, writeln, end).
- Remember their translations into Russian.
- Be able to name an analogue of the Russian word in English and vice versa.
- Remember the spelling of each of the English words (in small and capital letters).
- Remember the location of the English letters of the studied keywords on the keyboard.
- Formulate sustainable skills for quick typing of keywords.

All these problems are solved within the framework of two packages of tasks "Propaedeutics of words" (207 main tasks, 787 with leading tasks) and "Learning words (slowly)" (127 tasks, 1491 with leading tasks) (Dolinsky and Dolinskaya, 2018). At the same time, the course "Propaedeutics of words" presents tasks that do not require memorization. Each task has the necessary hints up to the location of the letter on the keyboard.

But the course "Learning words (slowly)" gradually helps the pupil to remember the order of letters in each word, their location on the keyboard, to train their typing. Moreover, the study of a new word ends with a control set of all learned words together.

As a result, almost all children cope with the task, although, of course, they do this with a significant difference in the time spent.

The propaedeutics of text programming is completed by studying the course "Number" (Dolinsky and Dolinskaya, 2019) containing 398 main tasks, along with 1244 auxiliary tasks. It begins with a slow transition from keywords to the first program (enter and display a number), presented below:

```
program p1;
var
  s : longint;
begin
  readln(s);
  writeln(s);
end.
```

And the ultimate goal of training in the "Number" course is to teach pupils to solve the first three problems of the 20 problems of the Olympiad in programming for pupils in grades 1–4 for input, formatted output, and simple number processing. The conditions for such tasks at the regional Olympiad held on April 20, 2018 are given below:

№ 1

Output Example:
(018)>(017)

№ 2

Input Example:
1
2
3

Output Example
1 plus 2 minus 3

Input Example:
5
1
4

Output Example
5 plus 1 minus 4

№ 3

Input Example:
2
3
8

Output Example
3+2=5
8−3=5

Input Example:
3
4
12

Output Example
4+3=7
12−5=7

Fig.1. Conditions of Tasks 1–3.

Developing Interesting Differentiated Training

The learning process is structured in such a way that, first of all, it is focused on the general development of the child: thinking, memory, independence, attention, hard work and creativity. Practice shows that far from all children continue study programming in secondary school, but certainly everyone gets the development. And the more they do the more development they get.

Since the child is engaged with us until he shows interest, we try to build the learning process as motivated as possible. This is done, first of all, due to the variety of presentation forms and ways of completing tasks. Another area is ensuring assignment of tasks for all pupils.

However, the feasibility of tasks for everyone with a linear system of building tasks leads to the loss of interest in classes for stronger children. That is why we have introduced differentiated training. In each topic, the main tasks are highlighted. They form the backbone of learning. To complete the package of tasks it is enough to complete only them. The strongest children can do so. However, if some main task is performed incorrectly, the pupil is automatically sent to the first facilitating task. Also he can get there if he presses the "I don't know" button. There can be several such assignments, gradually explaining how to complete the main task. Each of the facilitating tasks may have its own system of facilitating tasks, etc. Such a tree of teaching tasks provides a differentiated and adaptive approach, where each pupil can have his own educational trajectory, taking into account not only his general level of training, but also the current psychophysical state during a particular lesson. You're worse in thinking – you often ask for help, move more slowly on educational material. You think well – you ask for help less often, you move faster on educational material.

Task – Oriented Training

The training material is a set of tasks for the development of programs. If the child cannot solve the problem, he is offered a selection of tasks for the development of programs that gradually lead to the solution of this problem. If a new theory is required to solve the problem, it is given in the most concise and understandable form. If this is not enough, a lot of training tasks are offered for entering answers according to the initial data, compiling algorithms by permuting lines, composing permuting lines of programs according to given algorithms, a set of programs by algorithms, and many others.

Minimalist Approach to Theory

Theoretical information (Dolinsky, 2013) is given exactly in the quantity that is necessary for solving current problems.

To solve problems 4–10 of the Olympiad in programming in grades 1–4, you need to know the char, string data types, the concept of how to access the string character s at

position i (s [i]), as well as the built-in string processing functions length, copy, delete , pos. These data are entered by us in the learning process.

To solve the problems 11–15 of the Olympiad in programming in grades 1–4, you need to know the concept of a one-dimensional array, as well as the following standard algorithms for cyclic processing of one-dimensional arrays: summation of elements; counting elements with a given property (including complex conditions using AND / OR unions); determination of the maximum / minimum element and its number (all use the FOR loop); search for elements that have the specified property (using the WHILE loop operator).

To solve problem 16, one needs to be familiar with the concept of a two-dimensional array and its components (rows, columns, main and secondary diagonals) and be able to apply standard algorithms studied for one-dimensional arrays on a two-dimensional array and its components.

To solve problem 17, you must be able to find the distances: between two points; from one point to several; adjacent distances; all distances between two sets of points. And then apply the studied standard algorithms on one-dimensional and two-dimensional arrays.

To solve problem 18, you need to be able to come up with algorithms for processing strings of characters, practically without learning a new theory, with the exception of several new built-in procedures and functions (str, val, chr, ord, insert).

To solve problem 19, it is required to study and be able to apply any of the algorithms for sorting the elements of a one-dimensional array, for example, "bubble" or "exchange".

To solve problem 20, you need to be able to read a page of text – the conditions of the task and understand what exactly needs to be done (as a rule, apply one of the methods for standard processing of one-dimensional or two-dimensional arrays).

Regional Programming Olympiads for Pupils in Grades 1–4

It is very important that in the Gomel region 5 regional competitions are held for pupils of grades 1–4, which are held in October (school Olympiad), November (city Olympiad), March (school Olympiad) and April (two: city Olympiad and regional Olympiad). All of them have a standard format of 20 of the above tasks with a slight increase in complexity from the first to the fifth Olympiad. Since all competitions are held on the basis of the DL.GSU.BY website, anyone can take part in each of them. The official results at city and regional Olympiads include, of course, only official participants who decided the Olympiad under the control of the jury.

Training is organized in the direction of the Olympiads (or vice versa, the Olympiads correspond to the learning process). One way or another, the results of the Olympiads accurately show for each pupil which topics he has studied and which topics he has yet to study (or which topics he has learned poorly). And therefore, the Olympiads provide a strong motivating effect on pupils and teachers.

Preparation for the Olympiads of Grades 5–8

From the 2016–2017 academic year, the tasks 16–20 of the Olympiad for pupils in grades 1–4 are exactly the same as the tasks 3–7 of the Olympiad for pupils in grades 5–8. In addition, task 1 of the 5–8 grade Olympiad is a task on the topic "Introduction to Programming" and requires no more skills than the third task (processing the entered numbers and formatted output) of the 1–4 grade Olympiad. And the task 2 of the Olympiad of grades 5–8 is a problem on the topic of a one-dimensional array and, again, it does not require additional knowledge to solve it compared to the tasks of the 11–15 Olympiad for grades 1–4. Thus, it turns out that the best pupils of grades 1–4 can simultaneously participate in the Olympiad of grades 5–8 and not only participate, but also win diplomas in them, since in the Olympiad of 5–8 grades there are 10 problems, seven of which the best pupils can solve Grades 1–4. Thus, it turns out that students in grades 1–4, preparing for their olympiads, are simultaneously preparing for the olympiads that are waiting for them after moving to the 5th grade.

**Problems in School Mathematics, Olympiad Mathematics
and Informatics Mathematics**

On the DL.GSU.BY website there is a course "Mathematics", which contains tasks in mathematics in the program of grades 1–5, Olympiads in mathematics (Kangaroo; Beaver; Canadian, United Kingdom and Texas University Math Contests), informative math. Informatics math problems were obtained by automatically converting programming tasks into math problems, where according to the problem condition, for each of the proposed input data sets; the corresponding result must be manually calculated.

Continuing Education Motivation Contests

Obviously, the more time a pupil devotes to learning, the faster he will move on the training material, and the better his results at the Olympics. This is especially true in the case of an effective system of automatic differential learning, designed by us. In order to intensify the independent work of pupils at home, with his grandmother, in a sanatorium, etc. we hold Cups: "Autumn", "Winter", "Spring", "Summer", "Person of the Year". The main goal of this kind of competition is to determine who has solved the more problems for the fall, winter, spring, summer, for the whole academic year (from autumn to summer inclusive), respectively. For pupils in grades 1–4, such competitions are held separately for the courses "Informatics 2015" and "Mathematics". At the time of writing (August, 2019) in the academic year 2018–2019, more than 700 pupils have already taken part in the "Informatics 2015" competition, and more than 100 pupils have taken part in the "Mathematics" competition.

Many Years of Practical Experience

The pupils who were the first to study according to the system described above from the 1st grade, finished school in September 2018 (in Belarus it is 11 years of study). It is clear that annual operation and constant feedback led to the improvement in the training system.

Table 1 shows the number of pupils who studied in the competitions "Informatics" and "Mathematics" since the beginning of their conduct.

Table 1

Number of participants in the "Person of the Year" contest

	2011 /12	2012 /13	2013 /14	2014 /15	2015 /16	2016 /17	2017 /18	2018 /19
Informatics	249	301	144	166	335	655	760	755
Mathematics	-	146	43	118	131	144	257	135

Experience-based Training Scalability

On the basis of the DL.GSU.BY website, classes are conducted with primary school pupils by teachers from several educational institutions of Belarus, Russia, Armenia. I want to talk in more details about such work for younger schoolchildren in St. Petersburg. In the summer of 2017, the parent of one of the pupils who studied on our website, created a VKontakte group `https://vk.com/spb_dl.` The group has information about our system, invite to use DL.GSU.BY, serves for the experience exchange of parents, teachers and pupils.

At the time of writing of the article, there are already 4,150 subscribers. 1065 elementary pupils from St. Petersburg are taking the course "Informatics 2015". Along the way, obviously, according to the information received from this group, elementary school pupils from other cities of Russia began to study in the Informatics 2015 course: Tula, Chelyabinsk, Novokuznetsk, Mytishchi, Vologda, Ozersk.

Productivity

Children who study in the system described above on the basis of the DL.GSU.BY website regularly become graduates of city, regional, republican and international competitions (`http://dl.gsu.by/olymp/result.asp`).

Low Requirements for Teacher Professional Qualifications

The main advantage of teaching using the DL.GSU.BY website is automatic differentiated learning.

A teacher is required to:

- Open / close class.
- Be friendly to children.
- Maintain a working atmosphere in the classroom.
- Explain children how to work at home.

But, of course, if the teacher is interested in a better result, he himself will want to study, and will begin to think how to motivate / teach better.

"Accelerated Course 2013"

Initially, the package of tasks "Accelerated Course – 2013" was developed and implemented in the training course "Basic Programming", for pupils of grades 5–8. It contains assignments on 8 topics. The first 8 tasks in computer science Olympiads for pupils in grades 5–8: introduction to programming, one-dimensional array, two-dimensional array, geometry, strings, sorting, text problem, research. Each of the topics includes folders: "Technical minimum", "Olympiad 1–4 grades", "Olympiad 5–8 grades". This ensures the fastest possible advancement in the educational material of the most capable pupils.

In the learning process, it turned out that there are quite a few pupils in elementary school for whom the usual rate of learning slows down their development. Then "Accelerated Course – 2013" was copied from the course "Basic Programming" to the course "Informatics 2015". There are currently about 30 such pupils.

Support for the Transition to the Study of C ++

The system of automatic instruction in C++ programming (Dolinsky, 2017), in which all training tasks are generated on the fly based on the author's C++ solutions to the proposed problems, has been introduced on the DL.GSU.BY website since the summer of 2016. Practice has shown that the most natural and simple transition to C++ is carried out after studying the "Accelerated Course –2013", on the initiative and if the pupil wishes.

Conclusion

A system for teaching text programming in an elementary school, built on the basis of the DL.GSU.BY website at F. Skorina Gomel State University is described in this article. This system has the following advantages: zero entry threshold; propaedeutics

of text programming; developing, developing, interesting, differentiated training; task – oriented training; minimalist approach to theory; regional programming Olympiads for pupils in grades 1–4; preparation for the Olympiads of grades 5–8 (general tasks with the Olympiads of grades 5–8); problems in school mathematics, Olympiad mathematics and informatics mathematics; competitions motivating to permanent studies; many years of practical experience; experience-based training scalability; productivity; low requirements for professional qualifications of teachers; "Accelerated Course 2013" and support for the transition to the C ++ study.

References

Dolinsky M. (2013). An approach to teach introductory-level computer programming. *Olympiads in Informatics*, 7, 14–22.

Dolinsky M. (2014). Technology for the development of thinking of preschool children and primary school children. *Olympiads in Informatics*,8,63–68.

Dolinsky M. (2016). Gomel training school for Olympiads in Informatic. *Olympiads in Informatics*, 10, 237–247.

Dolinsky M. (2017). A New Generation Distance Learning System for Programming and Olympiads in Informatics

Dolinsky M.,Dolinskaya M (2018). How to Start Teaching Programming at Primary School. *Olympiads in Informatics*, 12, 13–24.

Dolinsky M.,Dolinskaya M (2019). Training In Writing The Simplest Programs From Early Ages, 13, 21–30.

Performance Statistics of Gomel pupils at international and national olympiads in informatics since 1997 up to 2019 (In Russian). `http://dl.gsu.by/olymp/result.asp`

M. Dolinsky is a lecturer in Gomel State University "Fr. Scoryna" 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. Scoryna" 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.

Consensus Algorithms for Highly Efficient, Decentralized, and Secure Blockchains

Diego I. ESTEVEZ

University of Waterloo
200 University Ave W, Waterloo, ON N2L 3G1, Canada
e-mail: destevez@uwaterloo.ca

Abstract. This paper about blockchain technology introduces its theory, implementation, and applications while focusing on the types of consensus algorithms. The methodology is linguistic and consists of a comparative analysis of the most popular algorithms. This paper is part of a broader effort to make these concepts more accessible and help develop an environment where students can grow and be part of this technological revolution.

Students with a background in algorithmic programming are uniquely suited to tackle highly impactful questions about the algorithms underpinning blockchains. After reading this paper, students should be able to build their own implementation of a blockchain and start doing research into this technology.

Keywords: blockchain, consensus algorithms, peer-to-peer networks, cryptocurrency.

1. Introduction

1.1. *Bitcoin as Leader of the Decentralized Revolution*

Bitcoin has been a very popular topic recently. It's been the promise of a technological revolution while simultaneously a rather controversial concept for governments and banks around the world. It introduced several concepts such as the blockchain and consensus algorithms, which enabled an effective scheme for decentralized ownership of information and transactions through a peer-to-peer network on the internet.

Bitcoin's core ideas can be summarized as transactions made between users without a middle-man and units of the currency directly owned by the individuals, not regulated by a central party like a bank or a company. The latter is a direct consequence of consensus algorithms, which, through cryptography and game theory, the network uses to verify the interactions and spread the changes across the network. Consensus algorithms

replace the need for trust between users and explicit coordination. In fact, in the words of Satoshi Nakamoto, the pseudonym of the inventor of Bitcoin, "What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party." (Nakamoto, 2008)

1.2. *What the Blockchain is*

The blockchain works as a public ledger that stores the interactions that users make within the network. As explained in the Bitcoin whitepaper by Nakamoto, in this system:

- Sending money between parties represents a change in the balance of the nodes involved. This change is referred to as a "transaction," and, as soon as it's made, it is broadcast to all the nodes.
- Members of the network, otherwise known as nodes or peers, store these transactions locally in a pool of unconfirmed transactions. Eventually, they will group these transactions into a block, a container of transactions, which, in Bitcoin's case, has a memory limit of 1MB.
- A blockchain contains a linear chain of blocks, with each of the blocks containing the cryptographic hash of the previous one. Thus, if any block prior to the current one is changed, its hash will change, and the chain of hashes will break.
- After collecting transactions to build a new block, a node has to confirm it and broadcast it to the rest of the nodes. To confirm the block, and the transactions contained in it, the node needs to meet the criteria of the consensus algorithm of the network. In Bitcoin's case, this algorithm is called Proof of Work (PoW) and asks for nodes to find a "nonce" (a number) that, when concatenated to some information in the block, leads to a hash with a specific number of leading zeroes. Therefore, since the node doesn't know which nonce will work beforehand, it needs to spend computational power by iterating through integers to find this number.
- After finding a nonce that works, nodes broadcast the block to the network along with the nonce. The rest of the nodes will receive this information and verify it. Verifying whether the nonce works is much easier than calculating it in the first place. Therefore, it is easy to check that computational power was spent by someone else.
- Bitcoin applies the principle that the longest chain will always be the "correct" one. This means that if the broadcasted block will lead to a longer chain and is verified, nodes will adopt it and add this new block to their blockchains. It's possible when two blocks are broadcasted at the same time that one block gets to some nodes before the other. Eventually, as nodes keep adding blocks, one of the chains will become the longest and will be adopted by the rest of the network.

The fact that the blocks are connected by their hashes allows for immutability. Additionally, anyone can join the network and get a copy of this immutable ledger, meaning that the information there is public and can't be censored.

Through consensus algorithms like PoW and the principle of the longest-chain, Nakamoto was able to solve the problem of double-expenditure, as well as circumventing some of the shortcoming of peer-to-peer networks, such as users/nodes going offline and the possibility of having nodes with different versions of the chain.

1.3. *Why Scalability is so Important*

In this paper's context, scalability means that the blockchain can handle a high transaction rate without clogging. This is key because, otherwise, it may be possible that applications of this technology will not be fully adopted. As a consequence, the capacity for a positive change of this technology in society will be undermined.

A network that fails to scale will negatively affect any participant's experience, thereby decreasing their retention and potentially incentivizing the use of worse alternatives that are useful in the short-run but centralized in the long-run.

2. The Problem

2.1. *Consensus Algorithms*

As previously explained, consensus algorithms are the mechanism that the network uses to confirm transactions. These algorithms pick the node that will add the next block based on a factor: with Proof of Work, a node with t percent of the total computational power of the network, known as hash rate, has a probability of t percent of finding the nonce and having his block confirmed. With other algorithms like Proof of Stake, a node, also known as validator in this context, is selected to add a block based on its total share of the blockchain's currency

As more transactions are made in the network in a short period of time, the size of the pool of unconfirmed transactions grows. Thus, transaction makers include a small amount of cryptocurrency, known as a transaction fee, as an incentive for validators to include the transaction in a block. This is due to the fact that when a validator confirms a block, it collects all the fees of the transactions contained within it. Therefore, the larger the transaction fee, the higher the incentive and the chance that it will be added soon in a block and confirmed.

Additionally, in Bitcoin's Proof of Work, a node is awarded newly generated currency by the system for every new block that it confirms. However, as time passes, these rewards are halved until they eventually converge to 0. At that point, the network's currency reaches the maximum supply that will ever be available. In Bitcoin's case, this number is about 21 million.

2.2. *The Difficulty of this Issue*

Because of the nature of these algorithms and peer-to-peer networks, there's usually a trade-off between scalability, security, and decentralization.

A consensus algorithm that scales is typically less secure since it is likely to be more fault-tolerant and have more relaxed requirements for validators. Higher validation standards results in more challenging tasks that delay the process.

Keeping a system totally decentralized also has an impact on performance since it requires more users to verify a single transaction and mitigate the chance of a node gaining an edge in the verification process and centralizing power. More complicated tasks by the consensus algorithms lead to a smaller set of nodes that can validate, concentrating power in them and taking away decentralization from the network.

An extremely secure system requires an extremely challenging task to confirm a block. Consequently, nodes need more incentives and resources to pursue this challenge of adding a block. Nodes with the most resources will have a higher chance of becoming successful validators. However, this tends to be a very small minority in very secure systems, which negatively affects the decentralization and scalability of the network.

In terms of security, blockchains in general tend to be vulnerable to 3 attacks: In the 51% attack, the malicious agent creates an unofficial copy of the network's blockchain. The agent then makes a transaction only in the official chain. For this transaction, the agent would have exchanged the cryptocurrency for something else. However, the problem arises when this user has so much computational power that, while the transaction still remains unconfirmed in the main chain, it can add blocks on its nonofficial chain faster than the rest of the network on the official chain. Thus, the nonofficial blockchain would eventually become larger than the official one, which, if broadcast, the rest of nodes would adopt based on the principle that the longest chain is the correct one. This means that the agent can now double-spend the money since the unofficial chain, which is the one that nodes would now have, does not contain the transaction that he or she had made. The double expenditure problem is called the Byzantine Generals problem.

Fig. 1. There's a trilemma of trade-offs between security, scalability, and decentralization in blockchains consensus algorithms.

Another famous attack is Sybil, where the malicious agent tries to fill the network with its own nodes to gain a larger total share. In PoW, the probability of mining the next block is equal to the share of the total computational power. Thus, the number of nodes that the agent controls is irrelevant. What matters is the share of the absolute CPU/GPU power.

Peer-to-peer networks are also vulnerable to denial of service attacks (DoS), where a node is bombarded with packets. In this case, the node gets flooded and cannot operate normally.

2.3. *The Purpose of the Question & Approach*

One of the most pressing questions is which consensus algorithm has the right balance between security, scalability, and decentralization for permissionless, open blockchains. In this qualitative paper, I try to answer that question by providing a linguistic framework to make a comparative analysis of the most popular algorithms that fall in this category.

3. The Implementation

To show how blockchains work, I will develop a basic version from scratch using Python. The snippets shown throughout the document are either in Python 3 or JSON.

Fig. 2 shows a reduced version of a Bitcoin transaction that highlights its most important features: the input amount, the fee for miners, a hash identifying the transaction, and the addresses of both the sender and the recipient. Nodes collect these transactions into their pools of unconfirmed transactions and then select some of these to go into their next block.

```json
{
    "hash":"8639c99fafd10ef8cd0b1c0499eb8983b4bc7810642589df374a0f87bb337ff4",
    "received-time":"2020-06-18 21:26",
    "input-amount":10.02440268,
    "fee":0.00006780,
    "sender":"1Ptv5qNTg6bpoMrH8zKqpiSA62jC3i76Nr",
    "recipient":"35EAYWQmq7nwBYqkYNLZKZf5WAY4sXs7BT"
}
```

Fig. 2. A simplified model of a Bitcoin transaction in JSON.

```json
{
    "hash":"00000000000000000001072a36c38d3c9e6cf1b3bc85d457606a39830574ba8c0".
    "previous-block":"53cc5f7efb064a603a1dca0ca9747c716ce4862e32e99f762a209b",
    "timestamp":"2020-06-18 22:54",
    "index":635362,
    "nonce":318525442,
    "transactions": {...}
}
```

Fig. 3. A simplified model of a Bitcoin block in JSON.

Fig. 3 is a simplified version of Bitcoin's Block 635362, which stores 2,386 transactions. Since each block contains the hash of the previous block, it would require an enormous power to change a value in a block prior and rebuild the chain. Thus, with every additional block, the chain is reinforced as it would require more power to modify the data.

The snippet in Fig. 4 contains a further simplified version of a transaction and block, as well as functions to instantiate the first block (genesis), add new blocks and transactions, and hash a block.

Since this blockchain is not part of any peer-to-peer network, it doesn't have any consensus algorithm or mechanism to prevent double-expenditure. However, several nodes need to have this chain and be able to synchronize it in real-time. Thus, next, we will see the most popular algorithms and implement Proof of Work, the most simple and common one.

```python
from time import time
import hashlib
import json

class Blockchain:
    def __init__(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.new_block(previous_hash='1', nonce=1) # Create the genesis block

    def new_block(self, nonce, previous_hash):
        block = {
            'index': len(self.chain) + 1,
            'timestamp': time(),
            'transactions': self.current_transactions,
            'nonce': nonce,
            'previous_hash': previous_hash or self.hash(self.chain[-1]),
        }

        # Reset the current list of transactions
        self.current_transactions = []
        self.chain.append(block)
        return block

    def new_transaction(self, sender, recipient, amount):
        self.current_transactions.append({ 'sender': sender,
                                           'recipient': recipient,
                                           'amount': amount})
        return self.last_block['index'] + 1

    @staticmethod
    def hash(block):
        block_string = json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(block_string).hexdigest()
```

Fig. 4. A Python 3 implementation of a basic blockchain[1].

---

[1] This implementation is based on that described in the article: Van Flymen, D. (2017, September 25). Learn Blockchains by Building One. From
https://medium.com/@vanflymen/learn-blockchains-by-building-one-117428612f46

4. Popular Algorithms

4.1. *Proof of Work*

Proof of Work (PoW) consists of combining the header of the block (that is, the part that includes the hash of the previous block and a Merkle tree) with the nonce to get a hash that meets a condition stipulated by the network. In Bitcoin's case, the task's complexity gets automatically adjusted every 2,016 blocks (or about 14 days) so that one block is confirmed, on average, every 10 minutes. To increase the difficulty, the network demands more zeroes in the hash, thereby exponentially increasing the computational power needed to confirm a block.

Since the first validator, known as miner in PoW, who discovers the nonce receives a reward (typically newly generated currency and all the transaction fees in the block), there's an incentive for nodes to compete to confirm the blocks as fast as possible. The node that can spend the most computational power has the highest chance of finding it first.

Some reasons why the difficulty of the task increases are to make up for advancements in technology that could make the task trivial and to decrease the chance that a nonce is found by chance. Additionally, the fact that finding the nonce requires so much 'work' means that it's hard for a single agent to gather 51% of the computational power of the network and be in the position to implement an attack.
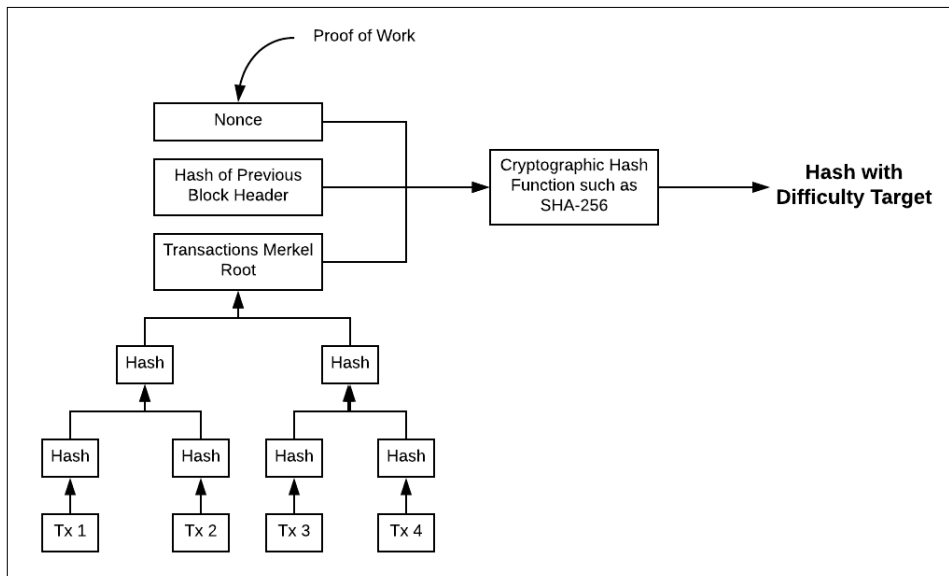
Fig. 5. Diagram illustrating the behaviour of the PoW algorithm, which consists of finding the cryptographic hash of a nonce, the hash of the previous block, and the transaction Merkle root and evaluating the resulting hash against a condition. (Kumar, 2018).

```
class Blockchain:
    ...
    @staticmethod
    def valid_nonce(last_nonce, nonce, last_hash):
        guess = f'{last_nonce}{nonce}{last_hash}'.encode()
        guess_hash = hashlib.sha256(guess).hexdigest()
        return guess_hash[:4] == "0000"

    def proof_of_work(self, last_block):
        last_nonce = last_block['nonce']
        last_hash = self.hash(last_block)

        nonce = 0
        while self.valid_nonce(last_nonce, nonce, last_hash) is False:
            nonce += 1
        return nonce
```

Fig. 6. The implementation of PoW into our basic blockchain model (van Flymen, 2017).

4.1.1. *The Implementation*

To implement PoW, we need the logic to compute nonces and verify them. We will continue using the example above (Fig. 6) that takes a further simplified approach to PoW.

In this fragment, three new functions are present:

- *Hash* computes the hash of a JSON block using SHA-256.
- *Valid_nonce* takes a suspected nonce, the last confirmed nonce, and the hash of the last block. Then, it joins them together, computes the hash, and checks whether the first 4 digits of the hash are all zeroes. This is a slightly different version of PoW than the one described previously, but the same principles still apply.
- *Proof_of_work* receives the hash and the nonce of the last block and looks for the nonce by iterating through a loop.

4.1.2. *Analysis*

Given that PoW requires so much computational power, it is a very safe consensus algorithm. For critical networks like Bitcoin, the probability of implementing a 51% attack is very low because it would require immense computational power. On the other hand, the chances of finding the nonce by sheer luck are also slim.

Scalability-wise, given the limit on block size and frequency, significant traffic increases (i.e., Bitcoin bubble in late 2017) lead to "bottlenecks" in the pool of unconfirmed transactions. This means that it might take many blocks (or days) until a specific transaction gets confirmed. Thus, the average transaction fee also must increase to incentivize miners to include the transaction in a block.

In PoW, decentralization is negatively impacted as the network grows. As the task's difficulty increases, it becomes more unaffordable for the average user to take part in the verification process because she or he might lack a powerful enough com-

puter to compete. The algorithm centralizes mining power in users who have high-end computers (ASICs mainly) and are located in countries where electricity is comparatively cheap.

The consumption of so much energy for PoW has a negative environmental impact, which cannot be ignored in the long run. In fact, Bitcoin alone (which implements PoW) uses the same amount of energy in a year as Denmark (Vashchuk & Shuwar, 2018). Besides, all this computational work is wasted: finding a nonce doesn't contribute or add to society in any meaningful way. Alternatives have been proposed that make use of this computational power to solve heavy computations in scientific research, thereby contributing to humanity (Wahab & Mehmood, 2018).

PoW is still a relevant algorithm that enabled many of the first applications of blockchain technologies (Bitcoin and Ethereum, to name a few) and provided a very secure method of verifying transactions. However, a system like PoW may be utterly inapplicable to applications that require transactions to be confirmed quickly and can scale to billions of users. A social network on the blockchain using PoW, for instance, would be unfeasible.

In the long-run, mainstream applications won't be able to use PoW because of the system's tendency for centralization, the lack of scalability, and the exponentially increasing costs of computation. PoW can still be very practical for small projects and as a means to generate the currency, but the community must realize that it will have to adopt another system if the application grows significantly. Furthermore, given that electricity won't be paid using the cryptocurrency, the currency's price will be negatively affected as mining costs increase.

4.1.3. *Task*

The best way to fully understand how a consensus algorithm works is by implementing one. At the beginning of the section, a basic model of a blockchain was shown using Python. Now, you can implement PoW using the information on this paper.

4.2. *Proof of Stake*

With PoW, a node has a probability of being selected to mine a block directly proportional to its share of the total computational power of the network. With Proof of Stake (PoS), computational power is replaced by the amount of the cryptocurrency, or tokens, that the node has. In most implementations of PoS, the user has to freeze some funds to be considered by the consensus algorithm as a possible validator, or minter in this context, where the higher these funds are and the longer they have been frozen, the higher the chance that the user is selected to add the next block. Other users have to attest that the block is correct and, if that's the case, the transaction fees of the block are distributed to the minter and the verifiers, and the frozen funds are returned. If the validator adds an invalid block, it is economically penalized by losing a part or all of his

or her frozen funds. Consequently, it is in the best interest of the wealthy nodes, which have the highest chance of being selected, to be honest, as being dishonest would have them lose a significant amount of money. For nodes to be considered as minters, they need to be online 24/7. Any computer with access to the internet can participate.

In PoS, the network takes a virtual approach to consensus. Instead of requiring a physical investment such as CPU/GPU power, the algorithm pseudo-randomly determines it based on a virtual investment, the amount of the native cryptocurrency that the node is willing to freeze. Thus, Proof of Stake is an energetically low-consumption alternative to Proof of Work and seeks to reward the nodes' commitment to the network.

In terms of decentralization, there's undoubtedly a propensity for wealthy nodes to get wealthier, as they would have a higher chance of being selected to validate a block and earning the transaction fees. The system can still be gamed by trying to hold as much currency for as long as possible, but the algorithm by slightly randomizing its choice for minter guarantees a minimum decentralization level and that the same agent will not always be chosen. Some networks, however, require a minimum amount to be staked, limiting an average person's capacity to become a minter. But, most of the time, this barrier tends to be smaller than buying high-end computers and paying significant electricity bills. Thus, there's a lower chance of disparaging economic inequality among the nodes.

Scalability-wise, PoS outperforms PoW. Furthermore, a system called sharding has been proposed that improves the scalability very significantly. In sharding, the network is split into different chains, on which validators work independently, and blocks are processed simultaneously.

However, a significant drawback of this algorithm is that, given that PoW doesn't award minters with newly generated currency like in PoW, there's no way for supply to be created of the cryptocurrency or token. Therefore, to produce the tokens of the blockchain, a variation of PoS should be implemented that also distributes these tokens fairly across the network. An option could be to use PoW to generate currency at the beginning and then switch to PoS once a number of tokens is met.

Unfortunately, pure PoS is less secure than PoW, as the incentive to not work on multiple chains is not present. In PoW, working on parallel chains would require a lot of computational power. In PoS, however, the nodes could simultaneously stake on multiple chains and increase the chance of being selected as minter. To solve this issue, called "nothing at stake," nodes that stake on various chains could be penalized by losing their frozen funds. Another vulnerability was exposed by Bitcoin researcher Andrew Poelstra, who showed that pure Proof of Stake (without any extra logic or procedures) is reversible, leading to the possibility of previous blocks being altered and enabling double-expenditure (Poelstra, 2015). Fortunately, the chance of a 51% attack is less realistic in PoS because the attack would require the agent to acquire a huge percentage of the tokens of the network. Based on economic supply and demand, each extra token purchased would become slightly more expensive. Thus, for blockchains worth billions of dollars, this would cost immensely.

4.3. *Delegated Proof of Stake*

Delegated Proof of Stake (DPoS) is a popular algorithm that consists on the community deciding on the nodes that will add the next blocks. The individuals who possess currency can vote for witnesses and delegates. Witnesses are the nodes that add blocks, and the network keeps a list of the nodes that receive the highest number of votes and picks the top X that will be adding the blocks. This number X changes by blockchain, and the network iterates through the top X witnesses giving each a few seconds to add its block. If the witness doesn't add the block in time, she or he gets penalized. Delegates, who are also voted on, check the validity of these nodes, ensure the network's well-being, and can propose modifications to the blockchain. In DPoS, there's continuous voting of delegates and witnesses.

DPoS positions itself as a system of decentralized governance. Holders of the cryptocurrency or token are the voters whose votes are weighted by how much currency they hold. To incentivize stakeholders to vote for them, witnesses have to share a percentage of the transaction fees they earn with their voters. As a consequence, witnesses have to appear trustworthy and that they will add many blocks and on time. On the other hand, if any party acts dishonestly, it can be immediately expelled by the voters.

This scheme removes the random factor of PoS and concentrates the validators among nodes with the highest credibility. The centralization parameter is trustworthiness, not wealth, which may be beneficial for the network because there's a competition to gain credibility among users. There are no significant barriers to entry, like investing in a high-end computer, and a node doesn't need capital to increase its chance of becoming a witness or delegate. This means that there's likely to be a higher degree of opportunity and economic equality among the nodes. Proponents of DPoS campaign that the fact that it is easy to enter the network means that the system is more decentralized than PoW or PoS. However, this forgets that voters need wealth to have some meaningful voting power. It is the wealthy users that have the most significant influence through their votes and those that essentially control the direction of the network.

As in PoS, the fact that the algorithm is not bound by a physical means or network's rules on block rate improves the capacity for scaling. But, opposite to PoS, the nodes don't have to be online all the time to participate and don't need to have the full chain. Additionally, the fact that minting power is concentrated in a few users significantly improves the rate at which blocks are added. Indeed, networks that select fewer witnesses will be the fastest, but this improvement in scalability will come at the direct cost of decentralization, and, by extension, security.

The fact that the age of coins is not taken into account is beneficial, as it removes the incentive for not moving wealth or trying to hack old, unused accounts that are poorly secured. Furthermore, the fact that voting power is proportional to wealth means that it would be extremely costly to get so much of it. Interestingly, while the "nothing at stake" problem is still present with this algorithm, if a user participates in multiple chains simultaneously, it will harm his or her reputation, essentially socially penalizing this user for the behavior.

For the entire system to work correctly, there needs to be an active, unorganized base of stakeholders that votes for the witnesses and delegates in the network's best interest. However, there is no guarantee of this. In fact, the tendency of the network towards centralization risks security, as it incentivizes the creation of cartels or groups to conspire together. For witnesses and delegates, this would be much more viable as there are few of them, and, by conspiring together, they could get away with malicious acts like double-spending. Networks that have a lower requirement of delegates to validate a block are especially vulnerable to one of these attacks, as it would be easier to meet that number. This phenomenon can also be extended to voters, who could conspire or be bribed to elect specific delegates and witnesses. There's certainly nothing that assures that the witnesses and delegates will never act maliciously.

4.3.1. *Solution as a Layer 2*

DPoS scales much better than PoS or PoW and can allow many applications that require several transactions per minute, such as blockchain video games (i.e., Crypto Kitties) or social networks (i.e., Steemit). However, DPoS should only be used for applications that can sacrifice security and decentralization for scalability. DPoS should not be the foundational consensus algorithm for systems that need very significant security and that must be trustless between nodes, such as those that manage financial transactions or health certificates.

An advantageous approach is to use DPoS as a Layer 2. In this scheme, a foundational blockchain runs a consensus algorithm like PoS or PoW and another blockchain linked to it, such as DPoS, serves to give scalability. Since PoW and PoS are relatively safe algorithms, there's reasonable level of integrity and security offered by the base chain. If the blockchain is attacked, the base chain can be used to revert to the last healthy block.

In my opinion, the community will inevitably have to resort to a Layer 2 solution in the long-run for many applications. Through hybrid protocols like this one, a stable network is achieved, where we can enjoy the benefits of the ultra scalability of DPoS while maintaining a sensible security level.

5. Conclusion

In my opinion, these algorithms serve different purposes, which, in large part, make it difficult to replace one by another.

Proof of Work is an algorithm that maximizes security to the point that is impossible with another consensus algorithm, and it is the easiest to maintain and implement. However, it becomes inefficient in the long run and cannot support a network like Bitcoin if the rate at which the number of users grows is maintained. Thus, I see PoW as an alternative that can serve to back financial systems and cryptocurrencies in the short/medium run, but it is a system limited by its own nature. Additionally, while it generates decentralization in the short term, it ultimately leads to centralization. By rewarding

clusters of high-performing computers, the system creates an incentive for the formation of cartels and groups that centralize block mining and validation.

Proof of Stake allows for long-term scalability by sacrificing some decentralization and security. There's likely to be less mobility of the tokens between the nodes as they are incentivized to freeze funds for as long as possible. The fact that wealthy nodes are likely to become wealthier is a risk for many cryptocurrencies, especially those that have a small market capitalization. For these blockchains, it would be cheap to buy a high percentage of the supply.

The Ethereum network, a blockchain for hosting decentralized applications (Dapps), is moving from PoW to Casper, their own implementation of PoS. Their objective is to increase the scalability of the network, which currently consists of 11,000 nodes and a $21 billion market capitalization; with PoW, the system is unable to sustain the Dapp ecosystem in the long run. Thus, the developers are willing to sacrifice some of the network's security and decentralization to achieve the project's purpose.

Delegated Proof of Stake is too risky for networks like Ethereum and Bitcoin that store so much value and still seek to maintain a general degree of decentralization. Nevertheless, DPoS can improve the scalability of blockchains by orders of magnitude, thereby enabling crucial applications for the growth of the blockchain ecosystem. Using a hybrid protocol that combines DPoS and PoW could be a very promising option for applications that still need security and scalability.

Indeed, each algorithm has its trade-offs and advantages and lies somewhere in a spectrum where there's no absolute "best." While more variations and new protocols will emerge, PoW will likely continue being the most popular option for its simplicity and security. On the other hand, PoS has more use-cases and seems to provide the most reasonable balance between decentralization, scalability, and security. It would not generate bottlenecks in the medium/short run, and security is usually not at high risk in most medium or small scale projects. Still, since it can't be used to generate the initial supply of the currency, another system like PoW will have to be used at the beginning.

With the information in this paper, students with a background in programming should have a clear picture of the state of the art of blockchain technologies and an understanding of the theory, implementation, and applications of these technologies.

In the future, problems related to blockchain architecture and algorithms could serve as an inspiration and a training resource for Olympiads because of their potential to be intellectually stimulating and to contribute to our knowledge and society.

References

De Quénetain, S. (2017). Delegated Proof of Stake: The crypto-democracy. Retrieved June 25, 2020, from http://www.blockchains-expert.com/en/delegated-proof-of-stake-the-crypto-democracy-2

Ferdous, M. S., Chowdhury, M. J. M., Hoque, M. A., & Colman, A. (2020). Blockchain consensus algorithms: A survey. ArXiv:2001.07091 [Cs]. Retrieved from http://arxiv.org/abs/2001.07091

Konstantopoulos, G. (2020). Understanding Blockchain Fundamentals, Part 2: Proof of Work & Proof of Stake. Retrieved June 25, 2020, from https://medium.com/loom-network/understanding-blockchain-fundamentals-part-2-proof-of-work-proof-of-stake-b6ae907c7edb

Kore, A. (2018). Building a blockchain. Retrieved June 25, 2020, from
    `https://medium.com/@akshaykore/building-a-blockchain-7579c53962dd`

Kumar, A. (2018). Fig. 2. Proof of Work in Bitcoin Blockchain [Digital image]. Retrieved June 26, 2020, from
    `www.vitalflux.com/bitcoin-blockchain-proof-work`

Larimer, D. (2017). DPOS Consensus Algorithm - The Missing White Paper. Retrieved June 25, 2020, from
    `http://www.steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-`
    `white-paper`

Li, C., & Palanisamy, B. (2020). Comparison of decentralization in dpos and pow blockchains. ArXiv:2002.02082
    [Cs]. Retrieved from `http://arxiv.org/abs/2002.02082`

Panda, S. S., Mohanta, B. K., Satapathy, U., Jena, D., Gountia, D., & Patra, T. K. (2019). Study of Blockchain
    Based Decentralized Consensus Algorithms. *TENCON 2019 - 2019 IEEE Region 10 Conference (TEN-
    CON)*. DOI:10.1109/tencon.2019.8929439

Poelstra, A. (2015). On Stake and Consensus.

Thin, W. Y., Dong, N., Bai, G., & Dong, J. S. (2018). Formal Analysis of a Proof-of-Stake Blockchain. *2018
    23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*. DOI:10.1109/
    iceccs2018.2018.00031

Van Flymen, D. (2017). Learn Blockchains by Building One. Retrieved June 26, 2020, from
    `https://medium.com/@vanflymen/learn-blockchains-by-building-one-117428612f46`

Vashchuk, O., & Shuwar, R. (2018). Pros and cons of consensus algorithm proof of stake. Difference in the net-
    work safety in proof of work and proof of stake. *Electronics and Information Technologies, 9*. DOI:10.30970/
    eli.9.106

Wagner, K., Keller, T., & Seiler, R. (2019). A Comparative Analysis Of Cryptocurrency Consensus Algo-
    rithms. *Proceedings of the 16th International Conference on Applied Computing 2019*. DOI:10.33965/
    ac2019_2019l2026

Wahab, A., & Mehmood, W. (2018). Survey of consensus protocols. ArXiv:1810.03357 [Cs]. Retrieved from
    `http://arxiv.org/abs/1810.03357`

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from
    `https://bitcoin.org/bitcoin.pdf`

D. Estevez is a 19-year-old Mathematical Physics student at the University of Waterloo. In the past, he founded a social network for citizen journalism, an NGO that was funded by Microsoft, Uber, and Dell, and sent capsules to the stratosphere that broke several records. He's also doing research at Democracy Mars into blockchain governance.

Recommending Tasks in Online Judges using Autoencoder Neural Networks

Paolo FANTOZZI[1,2], Luigi LAURA[1,3]

[1]*Italian Association for Informatics and Automatic Calculus (AICA), Italy*
[2]*Sapienza University of Rome, Italy*
[3]*International Telematic University Uninettuno, Italy*
e-mail: fantozzi@diag.uniroma1.it, luigi.laura@uninettunouniversity.net

Abstract. Programming contests such as International Olympiads in Informatics (IOI) and ACM International Collegiate Programming Contest (ICPC) are becoming increasingly popular in recent years. To train for these contests, there are several Online Judges available, in which users can test their skills against a usually large set of programming tasks.

In the literature, so far few papers have addressed the problem of recommending tasks in online judges. Most notably, as opposed with traditional Recommender Systems, since the learners improve their skills as they solve more problems, there is an intrinsic dynamic dimension that has to be considered: when recommending movies or books, it is likely that the preferences of the users are more or less stable, whilst in recommending tasks this does not hold true.

In order to help the learners, it is crucial to recommend them tasks that are challenging but not unsolvable compared with their current set of skills. In this paper we present a Recommender System (RS) for Online Judges based on an Autoencoder (Artificial) Neural Network (ANN).

We also discuss the results of an experimental evaluation of our approach in both the scenarios in which we consider, or not, the intrinsic dynamic dimension of the problem. The ANNs are trained with the dataset of all the submissions in the Italian National Online Judge, used to train students for the Italian Olympiads in Informatics.

Keywords: autoencoder neural networks, recommender systems, programming contests.

1. Introduction

Programming Contests (PCs) are competitions in which participants are faced a set of tasks that require writing computer programs. Recent literature have emphasized the importance and the effectiveness of PCs in the process of learning computer programming (Audrito *et al.*, 2012; Astrachan, 2004; Blumenstein *et al.*, 2008; Dagienė, 2010; Garcia-Mateos and Fernandez-Aleman, 2009; Wang *et al.*, 2011).

In order to train for PCs, learners use Online Judges (OJs), also known as Programming Online Judges, i.e., web based e-learning tools where a user can submit solutions
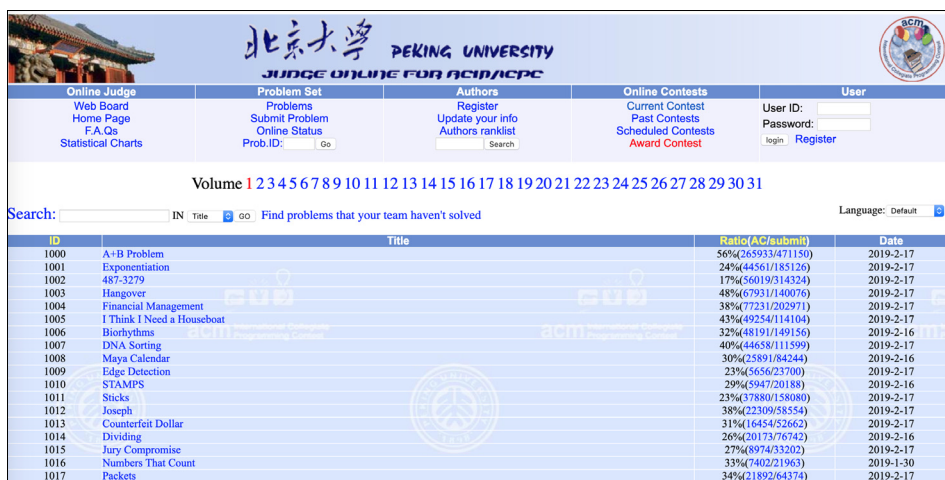
to a programming task. The user chooses a task from the many available; after reading its statement, that includes the required formatting of input and output or the use of a programming interface, the user writes a code to solve the task. The code is submitted to the OJ, that verifies both the correctness, usually by testing it against a certain number of test cases, and the efficiency, by checking that the running time and/or the memory usage is under some limit. In Fig. 2 is shown an example of a programming task.

However, choosing the right task is becoming a complex problem, and an example of an *information overloading scenario*, as observed in Yera Toledo *et al.* (2018): an unexperienced user has to choose from thousands programming tasks, many of which are probably beyond his current abilities. For example, University of Valladolid Online Judge has more than 200k users and 2k tasks, whilst SPOJ accounts approximately 600k users and 6k (public) tasks; in Fig. 1 we can see the list of available problems in the Peking University Online Judge `http://poj.org`.

With so many available tasks, it is important to help users selecting their next task by using a Recommender System (RS). Traditionally, RS are broadly divided into two categories: Content Based ones, in which the recommendations derive from features of the items to be suggested, and Collaborative Filtering approaches, in which the suggestion is based on the items chosen by users *similar to the current one*.

As observed in (Audrito *et al.*, 2019), there are some peculiarities of Online Judges that prevent the use of a general Recommender System:

- Users slowly improve their abilities, one task after the other, so the general concept of user *preferences* does not apply: recommending a movie or a novel differs significantly from recommending a task; a user will probably still like a novel after one year, whilst he might find a task too easy after the same amount of time.
- Users with *similar* skills, i.e. users to whom we might want to suggest the same set of tasks, might behave very differently in OJs, thus preventing us from considering them *similar*. For example, one might solve all the tasks involving a given

Fig. 1. The list of available problems in the Peking University OJ.

Abbatti S.p.A. (which is the Italian brand of *tearDown INC*) is a big enterprise that works in the field of tree felling. In particular, it's been a few years since it started improving in tearing down *barky trees*, a peculiar kind of trees which is very tall and thick. This particular species grow in a very tidy way: the woods made of these trees are actually a long horizontal line of trunks, placed at one decameter (32, 8 feet) one another. Each one of the trees has a particular height, which is expressed by a positive number (decameters).
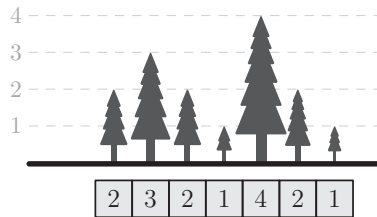
Tearing down one of these trees is a very hard thing to do and, although tearDown INC employes the most andnced technologies on the market, it is a very time consuming activity, since barky trees' bark is incredibly thick. The workers have the opportunity to chose in which direction (left or right) the tree should fall after the cut.

Each time a barky tree falls down it hits the trees that haven't been torn down which are placed on its falling trajectory; in other words, it tears down each tree which is closer than its height in the direction of the fall. Since the number of barky trees in this woods is huge this dynamic of the fall creates a domino effect.

In order to be the best enterprise in barky tree felling tearDown INC developed a system which is able to scan the whole wood, choosing which trees should be cut by workers and in which directions they should fall with the aim of cutting all the trees. It's important to recall that it's in the best interest of the enterprise to minimize the number of trees that need to be torn down by workers directly. Your role in this situation is to implement the system for tearDown INC.

Below we can see an example of a solution of the instance shown in the picture above: it is enough to cut two trees:
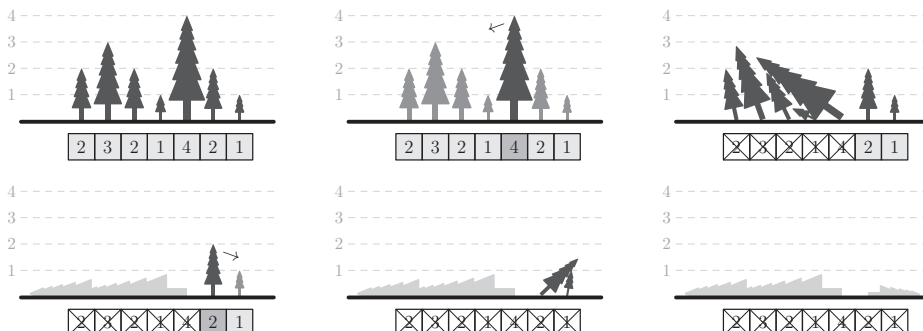
Fig. 2. An example of a problem from a programming contest; this task is taken from the final contest of the 2014 edition of the Italian Olympiads in Informatics (OII).

skill, while the other might just solve one task, related to that skill, and then move on to tasks involving different skills.

Most notably, as opposed with traditional Recommender Systems, since the learners improve their skills as they solve more problems, there is an intrinsic dynamic dimension that has to be considered: when recommending movies or books, it is likely that the preferences of the users are more or less stable, whilst in recommending tasks this does not hold true. In this paper we propose a task recommender system based on an Autoencoder Neural Network (ANN); in particular, we address both the *static* case, in which the user is represented by the task he solved, and the *dynamic* case, where we try to represent the growth of a user by the sequence of the problems he solved. For both cases we present a Recommender System (RS) for Online Judges based on an Autoencoder (Artificial) Neural Network (ANN). We trained and tested the ANN using data from the Online Judge used in the Italian Olympiads in Informatics (Olimpiadi Italiane di Informatica – OII) (Di Luigi *et al.*, 2016), targeted at secondary school students training. We compared our approaches against state of the art more classical recommender systems built using the Simple Python Recommendation System Engine (SurPRISE – `http://surpriselib.com`). The experimental results confirm the effectiveness of our approach.

Preliminary versions of this paper appeared in the Proceedings of the 17th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2020) (Fantozzi and Laura, 2020a) (the static case), and in the Proceedings 13th International Workshop on Social and Personal Computing for Web-Supported Learning Communities (SPeL 2020) (Fantozzi and Laura, 2020b) (the dynamic case); the comparison against state of the art more classical recommender systems has not appeared before.

This paper is organized as follows: the next section provides the necessary background related to programming contests, online judges, and recommender systems, whilst our approach is detailed in Section 3. In Section 4 we detail our experimental findings and concluding remarks are addressed in Section 5.

2. Related Works and Background

In this section we discuss related work and the necessary background concerning programming contests, online judges, and recommender systems.

2.1. *Programming Contests and Online Judges*

A programming contest is a competition in which contestants are faced with a set of programming tasks, also called problems, to be solved in a limited amount of time and/ or with a limited amount of memory usage.

A single task can be broken into different subtasks of increasing complexity: basic techniques might be enough to solve, within the given time and/or space limits, some of

the subtasks whilst the most difficult ones might require very specific algorithmic techniques and data structures. Popular programming contests are:

- The International Olympiads in Informatics (IOI), that are an annual programming competition for secondary school students patronized by UNESCO. `http://www.ioinformatics.org/`
- The ACM International Collegiate Programming Contest (ICPC) is a multitier, team-based, programming competition operating under the auspices of ACM. `https://icpc.baylor.edu/`
- The very recent International Olympiads in Informatics in Team (IOIT), that started in 2017, that are a team competition, like ACM ICPC, differently from IOI (individual competition). Currently there are only four nations involved: Italy, Romania, Russia, and Sweden. `https://ioi.team/`
- Google Code Jam, that is based on multiple online rounds that concludes in the World Finals. `https://code.google.com/codejam/`
- Facebook Hacker Cup, that is (citing from their site) "*an annual worldwide programming competition where hackers compete against each other for fame, fortune, glory and a shot at the coveted Hacker Cup*". `https://www.facebook.com/hackercup/`

The Online Judges are, usually, web based platforms that provide a large number of programming tasks to be solved. There are several popular OJ platform, we cite the already mentioned University of Valladolid Online Judge `https://uva.onlinejudge.org`, Sphere Online Judge (SPOJ) `https://www.spoj.com/`, CodeChef `https://www.codechef.com/`, and Peking University Online Judge `http://poj.org`.

Yera and Toledo (Yera Toledo *et al.*, 2018) present a brief survey on OJs, whilst more information on tools and techniques for automatic evaluation of solutions submitted to OJs can be found in (Ala-Mutka, 2005; Caiza and Del Alamo, 2013).

2.2. *Recommender Systems in OJs*

As already observed in the introduction, despite the large amount of literature devoted to RS, the peculiarities of recommendation in OJs, where the relation user-item is way more complex than the typical RS cases, prevent from using standard techniques and forces the development of ad-hoc methods. This aspect is detailed in the paper of Audrito *et al.* (2012), where the authors propose a first approach on building a RS by tackling the problem of ranking tasks in Online Judges.

Indeed, so far few research focused in the recommendation of tasks in OJs: we mention the traditional collaborative filtering method with a new similarity measure adapted to the case (Toledo and Mota, 2014), and an approach based on fuzzy logic (Yera Toledo *et al.*, 2018). Caro and Jimenez considered user-based and similarity-based approaches In (Caro-Martinez and Jimenez-Diaz, 2017). Di Mascio *et al.* proposed a framework that can allow recommendations and that can foster motivation in students by means of a lightweight, badge-based, gamified approach (Di Mascio *et al.*, 2018).
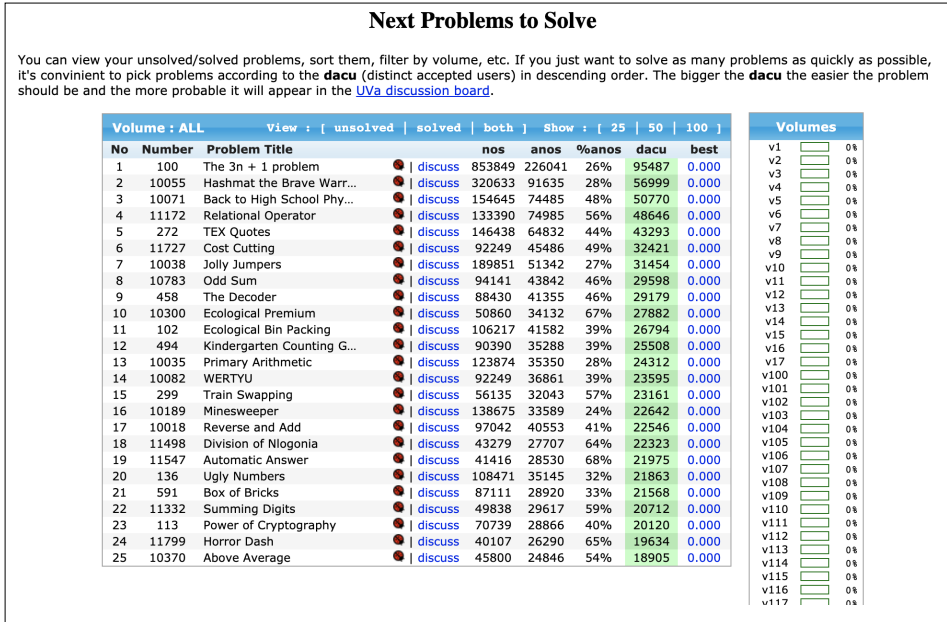
Fig. 3. The *Next Problem to Solve* section in the uHunt .

There is an online tool, developed by Stephen and Felix Halim, authors of the book Competitive Programming (Halim and Halim, 2013), called uHunt, that helps its users to choose the next problem to be solved, as shown in Fig. 3: their very practical (and effective approach) is to rank the problems according to their **dacu**, i.e. the distinct accepted users. Indeed, as they state, "*The bigger the dacu the easier the problem should be and the more probable it will appear in the UVa discussion board*".

The "classical approaches" that use counting to estimate the grade of difficulty have a well known drawback: the items suggested to the users will be always the popular ones that will become even more popular, and so even more recommended. This means that a new item will never be suggested.

2.3. *Recommender Systems and Artificial Neural Networks*

The use of deep learning techniques for recommender systems is divided in two categories that we call *classical* and *hybrid*.

The classical approach uses the standard architectures of neural networks, applying them to this task. So, in this case, the most important part consists in the formulation of the problem, since that, if the input data are not suitable to be the input of that specific deep learning technique, then the result will be totally inaccurate.

The other approach is the hybrid one, that consists in using more than one type of architecture at the same time. This kind of approach is useful when the input data are

not easily representable as a standard structure, like a user-item matrix. In (Zhang *et al.*, 2016) the authors use a Convolutional Neural Network to extract features from images and then an Autoencoder to build the recommender system on the features.

In (Zhang *et al.*, 2017a), in order to build a recommender system for hashtag in tweets, the authors use at the same time some CNNs and some Recurrent Neural Networks (RNNs). In this work they use the CNNs to extract features from the image and then the RNNs to extract information from the text, combining them using different weights based on co-attention.

Since that a recommendation task is similar to a dimensionality reduction task, many of the state-of-the-art techniques use some kind of Autoencoder to map the input in a smaller space, that will be the representation of the correlations in the recommender system. In particular, Sedhain *et al.* (2015) introduce the using of a vanilla Autoencoder to build a recommender system. They use a partial masked input (the same techniques we use in this work) and try to reconstruct it in output, the elements added in the output will be the recommended elements. Strub and Mary (2015) extend the work of Sedain *et al.* (2015): they use a denoising Autoencoder instead of the vanilla Autoencoder to build a more robust system.

Chen and de Rijke (Chen and de Rijke, 2018) follow a similar approach, but they use a Variational Autoencoder to perform top-N recommendation. They encode both the user ratings and some side information in the compact space in the Autoencoder. Zhang *et al.* (Zhang *et al.*, 2017b) generalize the Contractive Autoencoder paradigm into matrix factorization framework. Li *et al.* (Li *et al.*, 2015) combine a probabilistic matrix factorization with Marginalized Denoising Stacked Autoencoders to perform collaborative filtering. This work can be considered as a general framework to use these kinds of techniques; in this context, several works, including (Van den Oord *et al.*, 2013; Wang *et al.*, 2015; Wang and Wang, 2014), can be viewed as special cases of this framework.

3. Recommending Tasks Using Autoencoder Neural Network

Our goal is to provide recommendations to the users regarding the next task to deal with among all the tasks in a system. To build this kind of system we assume that:

- If a user obtains a score for a task it means that it is the max score possible for that user in that task.
- A user should solve the problems sorted by their grade of difficulty for the user; thus, a user should never try to solve a problem that is much harder than the last one he solved.
- It is possible to deduce the score for a new problem based on the scores the user obtained in other problems.

Note that there is one more assumption that is valid for the static case but not for the dynamic: given a snapshot of the scores for the users it is not important the order followed by the user to solve the tasks to foresee the score for another task.

Based on the above assumptions, we decided to build a model that takes as input the current scores of a user and provides probable scores for the same user for other tasks. Then we can choose between the forecasted scores and suggest the task with the highest score between them.

If we consider the score as a judgment of the user for the item (i.e, the programming task), then we can just exploit the already known techniques for recommending items. We chose to use an Autoencoder to build the model. Since that we want to use just the scores of the users, without any information from other sources, we take as input the scores for all the tasks and we differentiate between static and dynamic:

- Static: we mask a fraction of the scores in input as a non-solved task and we perform back- propagation from the complete scores.
- Dynamic: we use the scores of a fixed moment in time as input and we perform backpropagation from the next moment.

In this way, in the bottleneck layer, there should be a compact representation of the similarities of the tasks.

4. Experimental Evaluation

In this section we describe the results of our experimental evaluation. We distinguish the two approaches, i.e. static and dynamic, in the next sections, and then compare the results against the ones obtained using a state of the art more classical recommender system built using the Simple Python RecommendatIon System Engine (SurPRISE – `http://surpriselib.com`).

4.1. *Autoencoder Neural Networks: Static Case*

To test the method we have designed, we have taken the submission to the OII Training platform (Di Luigi *et al.*, 2016) in a defined time range. The submissions were in the form:

```
< user_id, task_id, datetime, score >
```

where each submission corresponds to a possible solution to a task from a user that performs a certain score, based on many test cases. We filtered out all the scores equal to zero because we can't know if they were just users testing the behaviour of the platform. Then we considered only the best score for each task, for each user, to ignore all the attempts to solve the problem before the user found the solution.

We performed a preliminary set of experiments with the original data: we built a user x task matrix where each cell contains the best score of the user for the task. The result is a 3148 x 409 matrix with 43051 non-empty cells. The matrix has an average of 105 submissions for each task and 13 for each user. The max number of users which have

submitted to the same task is 1070 and the max number of tasks with submissions from the same user is 336. We consider the zero valued cells as a problem with no submission from the user.

To use this matrix as a training set for this model, we duplicated the matrix and then we have randomly masked some positive scores with zero. The masked matrix will be the input to the model and the original matrix will be the output to reconstruct. The number of tasks masked for each user is a random number between 3 and 7, with the constraint that it should be anyway equal at most to the half of the submission for the user.

After the preliminary experiments, it was clear that data was too small, thus we performed an operation of data augmentation. in particular, we have repeated many times the same rows of the matrix with the result of a matrix with 8 times the rows of the original. Then each row has been randomly masked independently, so we unlikely had duplicated rows in the matrix. We have load all the data on a Google Colab instance with an available GPU. Then we have splitted the data on train and test set with a ratio of 0.8/0.2. We used Tensorow to build several models; the smallest was a Sequential model with 5 layers: the input layer, a dense 64 neurons layer, a dense 16 neurons layer, a dense 64 neurons layer, and an output layer with dimension equal to the input layer. All the activations for the layers are ReLU with a constraint of a max value of 1.0 (the max value of the score). We used an Adam optimizer with a learning rate of 0.001 and a mean squared error loss function.

We trained the model for 500 epochs with a batch size of 128, and we have imposed a validation split of 0.2. The resulting learning curve is the shown in Fig. 4, whilst the accuracy curve measured is depicted in Fig. 5.

The standard accuracy might be not fully representative of the error of the model (since that we have a sparse matrix), thus we computed a sum of the squared errors on each samples in the test set. The resulting values follow the distribution shown in Fig. 6; in Table 1 we report some stats of the SSE distribution.
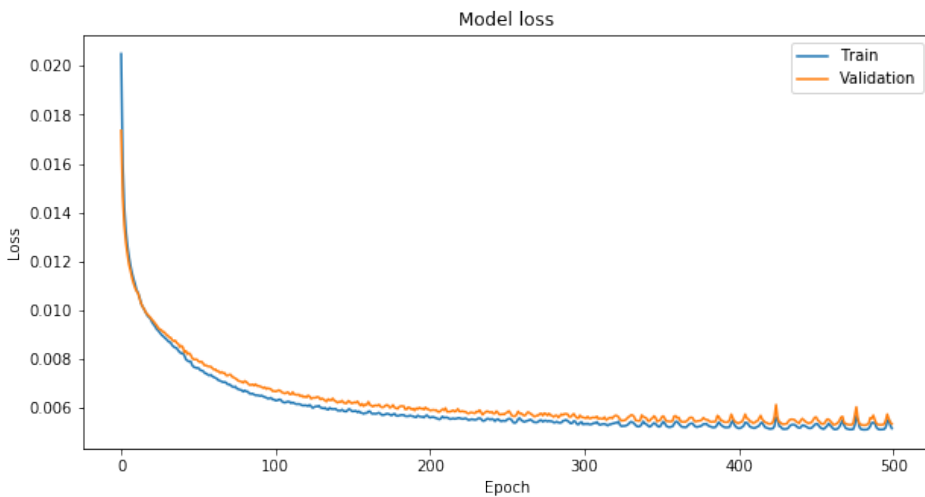
Fig. 4. Model loss – static case.
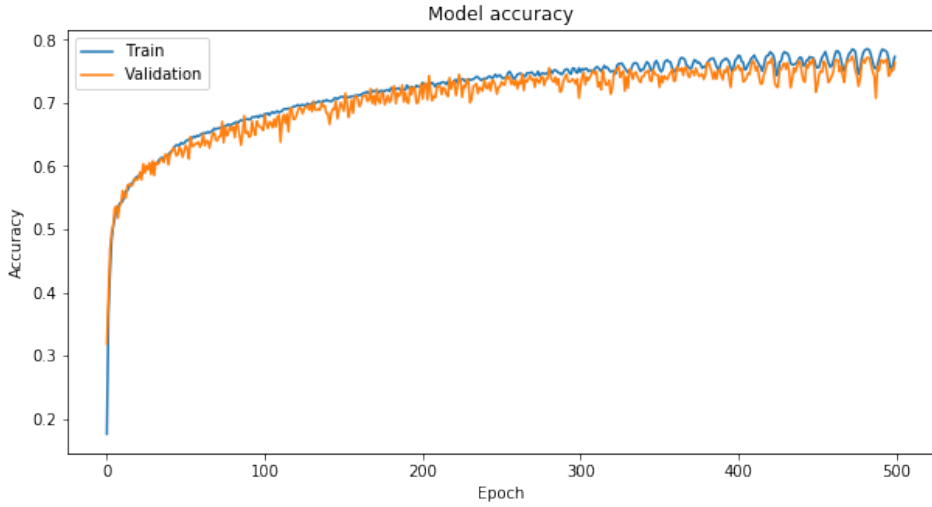
Fig. 5. Model accuracy – static case.
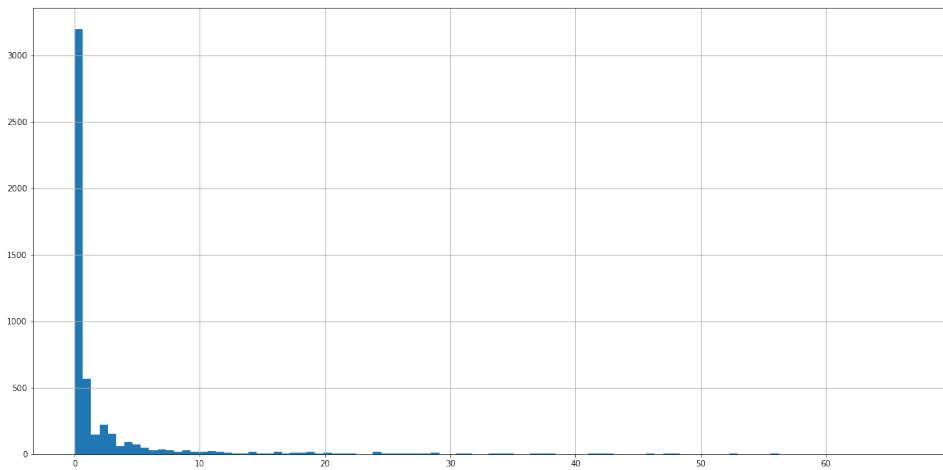
Fig. 6. Distribution of sum of squared errors (SSE) – static case.

Table 1

Statistics of the distribution of SSE (Fig. 9) – static case

mean	2.27
std deviation	6.07
min	0.00
25% (1st quartile)	0.00
50% (2nd quartile)	0.05
75% (3rd quartile)	1.38
max	66.19

4.2. *Autoencoder Neural Networks: Dynamic Case*

In the dynamic case, our interest was in the evolution of the users, so the dataset that we used was derived from the one, described in the previous section, for the static case. Indeed, from the baseline dataset, we built a user x task matrix where each cell contains the best score of the user for the task. The result is a 42155 x 409 matrix with 2035447 non-empty cells (we had to drop few submissions from the baseline dataset). As before, the dataset has an average of 105 submissions for each task and 13 for each user. The max number of users which have submitted to the same task is 1070 and the max number of tasks with submissions from the same user is 336. As before, we consider the zero cells as a problem with no submission from the user.

Since that the resulting matrix has many rows for each user (one row for each problem solved) that contains all the scores of the users until that moment, we use a row as input and we impose the next row for the same user as output. This means that we consider $n_i - 1$ samples for each user i, where n_i is the number of problems solved by the user i.

As for the static case, we load all the data on a Google Colab instance with an available GPU and the data was split into train and test set with a ratio of 0.8/0.2. We used Tensorow to build several models; the smallest was a Sequential model with 11 layers: the input layer, two dense 128 neurons layer, two dense 64 neurons layers, a dense 32 neurons layer, two dense 64 neurons layer, two dense 128 neurons layers, and an output layer with dimension equal to the input layer. Also in this dynamic case, all the activations for the layers are ReLU with a constraint of a max value of 1.0, and we used an Adam optimizer with a learning rate of 0.001 and a mean squared error loss function. We trained this model for 100 epochs with a batch size of 128, and we have imposed a validation split of 0.2.

The resulting learning curve is the shown in Fig. 7, whilst the accuracy curve measured is depicted in Fig. 8.
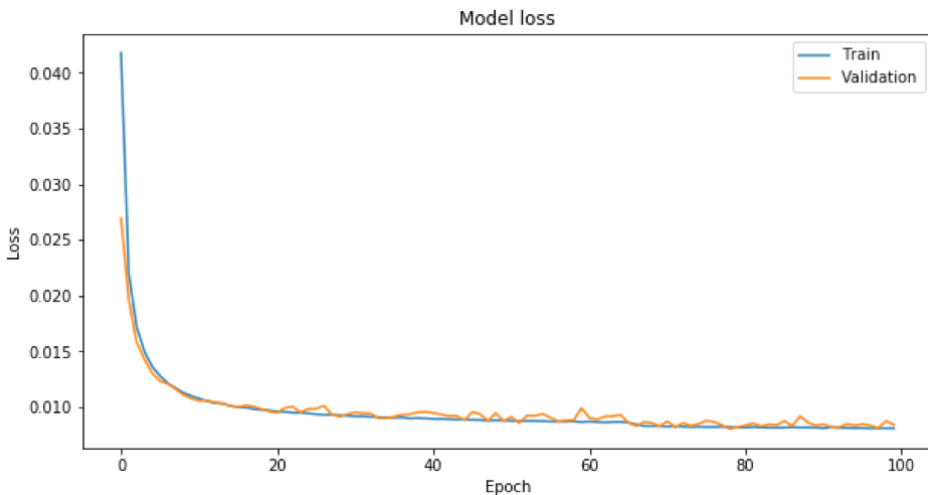
Fig. 7. Model loss – dynamic case.
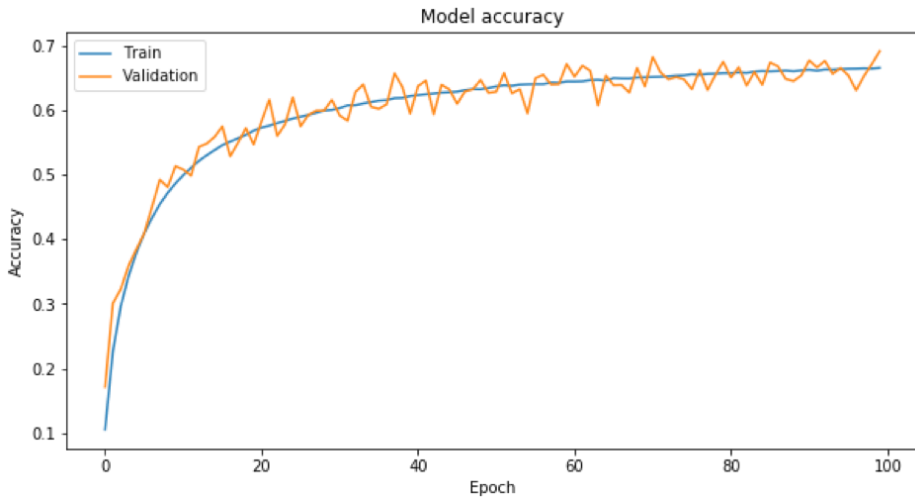
Fig. 8. Model accuracy – dynamic case.

Since that the standard accuracy doesn't represent well the error of the model (i.e., we have a sparse matrix) we computed a sum of the squared errors on each samples in the test set.

The resulting values follow the distribution shown in Fig. 9; in Table 2 we report some stats of the SSE distribution.

Overall, from the results shown above, it seems that the static approach, described in the previous section, seems to perform better than the dynamic one; this might be due to the way the dataset has been built, and we plan to compare the two approaches against other different datasets.
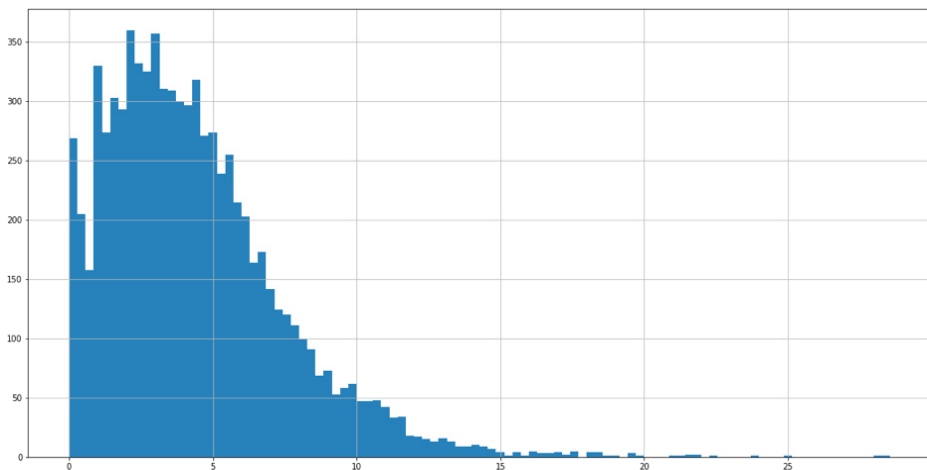
Fig. 9. Distribution of sum of squared errors (SSE) – dynamic case.

Table 2

Statistics of the distribution of SSE (Fig. 9) – dynamic case

mean	4.40
std deviation	3.10
min	0.00
25% (1st quartile)	2.12
50% (2nd quartile)	3.86
75% (3rd quartile)	5.98
max	28.56

4.3. *Comparison Against Classical Recommender Systems*

In this section we compare our approaches against state of the art Recommender Systems built using the python SurPRISE library: we used 11 models from this library. We trained these models using the static dataset, i.e. the original dataset; the dynamic dataset derives from this one and the dynamic ANNs, as seen in the previous section, performed not as well as the ones trained on the static dataset.

Indeed, we experimented with 32 different ANNs for the static case and, after evaluating the best performers, we experimented with 8 different ANNs for the dynamic case. In Table 3 we can see the mean square error (MSE) for all 11 SurPRISE models,

Table 3

The Mean Square Error of all the tested model, sorted from the biggest (worst) to the smallest (best). Here we compare all the models from the SurPRISE library against the three best performers of the several Autoencoder Neural Network we tested

Model	MSE
NormalPredictor	0.1004863793
CoClustering	0.0903564508
SlopeOne	0.0599415086
NMF	0.0545891728
KNNBasic	0.0544956720
KNNWithZScore	0.0528847039
KNNWithMeans	0.0526918100
SVD	0.0512103173
BaselineOnly	0.0484940726
SVDpp	0.0479013953
KNNBaseline	0.0478215959
(dynamic) autoencoder-plus-time-512-128-Dropout(0.1)-512-lr0.001	0.0031555248
(dynamic) autoencoder-plus-time-2048-512-Dropout(0.1)-2048-lr0.001	0.0025685430
(dynamic) autoencoder-plus-time-1024-256-Dropout(0.1)-1024-lr0.001	0.0025619145
(static) autoencoder-2048-512-Dropout(0.1)-2048-lr0.0005	0.0000349358
(static) autoencoder-1024-256-Dropout(0.1)-1024-lr0.0001	0.0000281454
(static) autoencoder-2048-512-Dropout(0.1)-2048-lr0.0001	0.0000191324

and for the three best ANNs for both the static and the dynamic case. The results shown in the table are sorted from the biggest (worst) to the smallest (best). It seems that, at least for this dataset, the ANNs outperform each model from the SurPRISE library. In the table, the three best results belong to ANNs trained for the static case, but in all our experiments, i.e. 40 different ANNs (32 for the static case and 8 for the dynamic case), we did not observe such a clear separation between the ANNs trained with the two different datasets.

5. Conclusions

In this paper we proposed the design of a recommender system for tasks suggestions in Online Judges, based on a Autoencoder Neural Network. We trained the ANN with the data from the OJ used by the secondary school students training for the Italian Olympiads in Informatics (Olimpiadi Italiane di Informatica – OII) (Di Luigi *et al.*, 2016; Di Luigi *et al.*, 2018).

We tested two different approaches: a *static* one, that is more typical of a recommender system, and a *dynamic* one, in which the dataset has been modified in order to explicitly represent the evolution of a user. We also compared our approaches against more traditional Recommender Systems model built using the python SurPRISE library.

We definitely think that Online Judges deserve their specific recommender systems, and we hope that our one is a first step to the development of such systems. We plan to implement our approach inside the italian OJ, and we are available to collaborate with the developers of other Online Judge systems, by either implementing RS into those systems or by testing our models against other datasets.

References

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2), 83–102.

Astrachan, O. (2004). Non-competitive programming contest problems as the basis for just-in-time teaching. In: *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pages T3H/20–T3H/24 Vol. 1.

Audrito, G., Demo, G. B., and Giovannetti, E. (2012). The role of contests in changing informatics education: A local view. *Olympiads in Informatics*, 6.

Audrito, G., Mascio, T. D., Fantozzi, P., Laura, L., Martini, G., Nanni, U., and Temperini, M. (2019). Recommending tasks in online judges. In: *Methodologies and Intelligent Systems for Technology Enhanced Learning, 9th International Conference, MIS4TEL 2019, Avila, Spain, 26–28 June, 2019*, volume 1007 of *Advances in Intelligent Systems and Computing*, pages 129–136. Springer.

Blumenstein, M., Green, S., Fogelman, S., Nguyen, A., and Muthukkumarasamy, V. (2008). Performance analysis of game: a generic automated marking environment. *Computers and Education*, 50, 1203–1216.

Caiza, J. and Del Alamo, J. (2013). Programming assignments automatic grading: Review of tools and implementations. In: *INTED2013 Proceedings*, *7th International Technology, Education and Development Conference*, pages 5691–5700. IATED.

Caro-Martinez, M. and Jimenez-Diaz, G. (2017). Similar Users or Similar Items? Comparing Similarity-Based Approaches for Recommender Systems in Online Judges. In: Aha, D. W. and Lieber, J., editors, *Case-Based Reasoning Research and Development*, volume 10339, pages 92–107. Springer International Publishing, Cham.

Chen, Y. and de Rijke, M. (2018). A collective variational autoencoder for top-n recommendation with side information. In: *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, pages 3–9.

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

Di Luigi, W., Fantozzi, P., Laura, L., Martini, G., Morassutto, E., Ostuni, D., Piccardo, G., and Versari, L. (2018). Learning analytics in competitive programming training systems. In: *2018 22nd International Conference Information Visualisation (IV)*, pages 321–325.

Di Luigi, W., Farina, G., Laura, L., Nanni, U., Temperini, M., and Versari, L. (2016). oii-web: an interactive online programming contest training system. *Olympiads in Informatics*, 10, 195–205.

Di Mascio, T., Laura, L., and Temperini, M. (2018). A framework for personalized competitive programming training. In: *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–8.

Fantozzi, P. and Laura, L. (2020a). Collaborative recommendations in online judges using autoencoder neural networks. In: *Proceedings of the 17th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2020)*.

Fantozzi, P. and Laura, L. (2020b). A dynamic recommender system for online judges based on autoencoder neural networks. In: *13th International Workshop on Social and Personal Computing for Web-Supported Learning Communities (SPeL 2020)*.

Garcia-Mateos, G. and Fernandez-Aleman, J. L. (2009). Make learning fun with programming contests. In: *Transactions on Edutainment II*, pages 246–257. Springer.

Halim, S. and Halim, F. (2013). *Competitive Programming, Third Edition*. Lulu. com.

Li, S., Kawale, J., and Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820.

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In: *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112.

Strub, F. and Mary, J. (2015). Collaborative filtering with stacked denoising autoencoders and sparse inputs.

Toledo, R. Y. and Mota, Y. C. (2014). An e-learning collaborative filtering approach to suggest problems to solve in programming online judges. *Int. J. Distance Educ. Technol.*, 12(2), 51–65.

Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In: *Advances in Neural Information Processing Systems*, pages 2643–2651.

Wang, H., Wang, N., and Yeung, D.-Y. (2015). Collaborative deep learning for recommender systems. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244.

Wang, T., Su, X.and Ma, P., Wang, Y., and Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. *Computers and Education*, 56, 220–226.

Wang, X. and Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In: *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 627–636.

Yera Toledo, R., Caballero Mota, Y., and Martínez, L. (2018). A Recommender System for Programming Online Judges Using Fuzzy Information Modeling. *Informatics*, 5(2), 17.

Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 353–362.

Zhang, Q., Wang, J., Huang, H., Huang, X., and Gong, Y. (2017a). Hashtag recommendation for multimodal microblog using co-attention network. In: *IJCAI*, pages 3420–3426.

Zhang, S., Yao, L., and Xu, X. (2017b). Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 957–960, New York, NY, USA. Association for Computing Machinery.

P. Fantozzi is involved in the training of the Italian team for the IOI since 2018. He is a Ph.D. student in Engineering in Computer Science at "Sapienza" University of Rome. He is lecturer at LUISS University for the courses: Lab of computer skills, Customer intelligence and big data analysis logics, Introduction to network science.

L. Laura is Associate Professor at Uninettuno university; he 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.

Operator Utilization and Abstract Conceptions

David GINAT

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

Abstract. Algorithmic challenges occasionally embed the utilization of specified operators. Suitable operator utilization is tied to recognition and capitalization on its characteristics, which should be unfolded and comprehended. In seeking comprehension, one may invoke abstraction perspectives with which to view the operators' concrete features. We display invocations of such perspectives in problem solving of several "unplugged" challenges, and mention our experience with students. Problem solving of such challenges elaborates on comprehension and verification. It enhances conceptual practice, without programming considerations. Practice of the interplay between the abstract and the concrete elevates problem solving competence and confidence.

Keywords: abstraction, problem solving.

1. Introduction

Abstraction is a primary notion in computer science (CS). Common themes of abstraction include top down, design patterns, abstract data types, divide & conquer, recursion, and more. They are related to design, problem solving, and the mixture of both. Problem solving abstractions are relevant at various levels in solving algorithmic challenges (Wing, 2006; Ginat and Blau, 2017). Problem solvers may need to think concurrently at multiple levels of abstraction and "move" back and forth between the concrete and the conceptual. The concrete involves direct consideration of the givens of a problem to solve. The conceptual may encapsulate a perspective of *hiding, or ignoring details* and a perspective of *relating to properties of recognizable parts* (Frorer *et al.*, 1997). Algorithmic problem solvers, including IOI contestants, should invoke perspectives of both kinds.

One element in various IOI tasks is that of operator utilization. An operation or function is specified, and contestants are requested to repeatedly use it, in a suitable and efficient way. Some examples are the task Sorting a Three-Valued Sequence in IOI 1996; the task Median Strength in IOI 2000; and the task XOR in IOI 2002. In the 1996 task, the given operator was a simple exchange operation; in the 2000 task, the given operator

was a function that finds the median of three different values; and in the 2002 task, the given operator was an invert operation (called xor) on a rectangle of black/white pixels in a matrix. In all the three tasks, problem solvers had to relate to properties of the given operator, and capitalize on its characteristics. They also had to ignore details, while focusing on particular parts (e.g., focusing on "imagecorners" in the 2002 task).

All the three operators in the above tasks are simple operators. In addition, the task goals are simply defined (though not easily reached). The solutions require significant insight, but do not require complex computational structures. Correctness verification is fundamental; and efficiency consideration appear in various forms – optimality (in the 1996 task), various time-complexity levels (in the 2000 task), (Horváth and Verhoeff, 2002), and lower bound measures (in the 2002 task).

Operator utilization may be practiced. Suitable practice may enhance experience and familiarity with recognizing, and capitalizing on operator characteristics. Such practice may reduce gaps between abstract perspectives and concrete implementations, and strengthen operator utilization conceptions. The reduction of gaps between one and her object of thought may be a means for enhancing abstraction capabilities (Hazzan, 2002).

Practice of operator utilization may be offered with programming, as in the above IOI tasks. It may also be conducted "unplugged" (Fellows *et al.*, 2005). The former involves conceptual notions combined with applied implementation. The latter may exclude implementation considerations, while solely focusing on relevant abstraction conceptions. This may occur with exploration of hidden patterns and properties of operators, as well as with proof considerations. The tasks may be of various levels of difficulty. A tutor may conduct gradual practice and examination of her students' conceptual competencies, elaborate on abstraction perspectives, and develop student awareness and capabilities.

In this paper we demonstrate "unplugged" practices of operator utilization, with several examples that involve simple operators – skip, reverse, and xor. The operators' utilizations are requested in tasks that involve both algorithmic design and proof consideration. The tasks and their solutions are displayed in the next section. We embedded these tasks in-between programming challenges, at the beginning of the training of our top 30 students (before reaching the IOI level). In the last section we relate to abstraction facets of these tasks, elaborate on their role, and indicate our experience with students.

2. Unplugged Operator Utilizations

The tasks in this section involve rather simple algorithmic schemes, but not necessarily trivial algorithmic properties. In some of the tasks the goal may not be achieved for every input, or initial state. The problem solver should recognize the initial states for which the goal is attainable and the initial states for which it is unattainable. For the former case, an algorithm should be developed, and for the latter case verification of

unattainability should be devised. In another task the goal may always be reached, but a minimal number of operator invocations is required. An argument of minimality should be formulated. Both of the algorithmic schemes and the sound argumentations are based on recognized properties, and involve the abstraction perspectives of relating to properties of recognizable parts and ignoring subordinate details.

Some tasks are rather simple, and some are more challenging. The tasks are short, and different from the IOI tasks mentioned above. Yet, the considerations necessary in their solutions contribute to the practice of abstract conceptions. The first two tasks involve sorting.

> **Skipping pairs.** The operator `skip(i,j)`, 0<i,j<N, skips the two adjacent integers, in the locations `i`, `i+1` in a list of integers, into the locations `j`, `j+1` (respectively). Given a random permutation of the integers 1..N, N>10, sort the permutation using the given operator, or output "Sorting is impossible".

The task specification hints that there may be inputs for which sorting may not be obtained. One sorting scheme may be based on the following: skip the integer 1, together with the integer next to it to the beginning of the permutation; then skip the integer 2; then – the integer 3; and so on. If at the end of this process the rightmost two integers are ordered, then we are done. But what if they are not in order? Can we correct this situation by additional invocations of `skip`? Perhaps there is a better algorithmic scheme? We relate to relevant ordering properties.

The situation in which two integers are not in order is called *inversion* (Knuth, 1973). The sorting goal – an ordered permutation – involves 0 inversions. Two properties that are relevant to examine are: 1. The number of inversions in the initial permutation; and 2. The change in the number of inversions by the operator. In examining the latter, we examine "what happens" when we skip a pair of integers over a third integer, say from right to left. If both of the integers are greater than the third, then the change in the number of inversions is +2; if they are smaller than the third, then the change is -2; if one is smaller and one is greater, then the change is 0. This yields the following *parity property*:

> *In every invocation of `skip`, the change in the number of inversions is an <u>even</u> number (regardless of the skip length). Thus, the initial parity of the number of inversions never changes.*

The above observation implies that if the initial number of inversions is odd, then the goal cannot be obtained, since the number of inversions may never reach 0. Thus, if the rightmost pair of integers is unordered at the end of the skipping process, then sorting cannot be attained. All in all, the recognition of the unattainable cases involved the *operator's property – no change in the parity of the number of inversions*. The next task involves sorting with a different operator.

Reversing 3 and 4 tuples.
A. The operator `reverse3(i)`, 1≤i≤N-2, reverses the order of three adjacent integers in a list, the left one of them being in location `i`. Given a random permutation

of the integers 1..N, N>100, sort the permutation using the given operator, or output "Sorting is impossible".

B. Answer part-A when the operator is `reverse4(i)`, 1≤i≤N-3, which reverses the order of four adjacent integers.

In part-A, we may immediately notice that the middle element serves as an "axis", and its location does not change in an application of the operator. The two end elements are swapped. This implies that the operator does not change the <u>parity of the locations</u> of the three elements. Therefore, elements in the odd locations will always remain in the odd locations, and so is the case with elements in the even locations. The operator will yield sorting if and only if all the odd integers are initially in the odd locations.

In part-B no element serves as an axis, and each of the four elements changes the parity of its location. In seeking relevant properties on which to capitalize, it may be beneficial to simplify the task and initially focus on a particular subset of the input. One such subset may be that of reversed permutations that should be inverted. Upon examining short cases of such permutations, one may notice that the cases of N=4,5,8,9 may be sorted (inverted), and the cases of N=6,7.10,11 are problematic. In the cases of N=8,9 one may use N=4,5 as generic templates that will be repeatedly used, in a *rolling* scheme, in which integers will be rolled to their desired destinations. This may be extended for larger values of N for which sorting is possible.

Why do some cases pose difficulties? We may seek the solution by relating to the notion of the number of inversions, as in the first task. In the reversed permutations of lengths N=6,7,10,11, the number of inversions is odd, while in N=4,5,8,9 it is even. This leads us to the change in the parity of the number of inversions of the operator. Indeed, as in the first task, here too, the <u>parity</u> of the number of inversions is <u>not changed</u> by the operator. This *property of the operator* paves the way for explaining unattainability of half of the initially-inverted cases.

At this stage we may widen the range of cases, and examine the general case. As in part-A, rolling may be a useful scheme for "bringing" each of the permutation elements to its final destination. However, as in the previous task, some rightmost elements may not be in order at the end of rolling. If the parity of the number of inversions is initially odd, then obviously sorting may not be attained.

But what if that parity is even among the last four integers, and they are still not ordered? Does this mean unattainability? Perhaps not. Perhaps a more involved ordering scheme is needed? Further abstract conceptions of ordering may shed additional light. We leave the answers to the interested reader.

At this stage we turn to two different tasks with the operator `xor`. In the first task, the focus is on optimality, with respect to the number of operator invocations. In the second task unattainability is relevant again.

Uniform color 1.
A. The cells in an N-cell row (N>100) are randomly colored black and white. The operator `xor1D(i,j)`, 1≤I,j≤N, inverts the color of each of the cells between the `i`-th and the `j`-th cells (including `i` and `j`). Transform the whole row into white with a minimal number of operator invocations.

B. An <u>additional</u> operator is provided – `xor2D(i,j)`, 1≤I,j≤N, which inverts the color of each of the cells between the `i`-th and the `j`-th cells in both of the rows. Transform the whole matrix into white with a minimal number of invocations.

In part-A, the elements on which to focus are the vertical sides that separate between black and white cells, and between black end-cells and the "out". We name the total number of these sides `S`. Notice that `S` is always even. The goal is to reduce it to 0. A single employment of `xor1D` may reduce `S` <u>by at most 2</u>, since the relevant impact of `xor1D` is only on the end sides of a given rectangle, and not on the inner sides. An observation of *ignoring details* is that the lengths and locations of chosen rectangles are <u>unimportant</u>, as long as the two ends of each rectangle are sides that separate between black and white. The total number of invocations is `S/2`. It is the minimum.

In part-B, rectangles' dimensions may be 1D or 2D. Below is an example matrix.

Should we consider not only vertical sides, but also horizontal sides? Not quite. The change in the horizontal sides derives from the change in the vertical sides. The horizontal sides may be *ignored*. We may view `xor2D` as reducing `S` <u>by at most 4</u> vertical sides that separate between black and white.

The algorithmic solution will be divided into two stages – a stage of employing `xor2D` and a stage of employing `xor1D`. The operator `xor2D` will be applied as long as it can reduce `S` by exactly 4 in each invocation. Then `xor1D` will be applied on the remaining sides that separate vertically between black and white.

Can `S` be reduced by 3? In addition, should we specify how we choose a rectangle for an application of the operator `xor2D`, among several choices? The answers to these questions should be part of an argument for minimality of the number of operator invocations. If the answers to these questions are "no", then we may argue for minimality, as each stage of the two stages of the algorithmic solution above would involve a minimal number of steps

Uniform color 2.
A. The cells in an N×M matrix (N,M>100) are randomly colored black and white. The operator `xor2(i,j,d)`, 1≤i≤N, 1≤j≤M, inverts the colors of the cell <i,j> and its adjacent cell in direction `d`, where `d` is one of four values – l, r, u, d (left, right, up, down). Transform the whole matrix into white, or output "Transformation is impossible".
B. Answer part-A when the operator `xorL` is provided <u>instead of</u> `xor2`. `XorL(i,j,d1,d2)` inverts the colors of 3 cells in an ⌐ or ⌐ shape – the cell <i,j>, and its adjacent cells in the directions `d1` and `d2`.
C. Answer part-A when the operator `xor3` is provided <u>instead of</u> the previous two. `Xor3(i,j,d)`, 1≤i≤N, 1≤j≤M, inverts the colors of: cell <i,j>, its adjacent cell in direction `d`, and the next cell in direction `d`.

In part-A We start by trying to whiten the matrix in a "snake-like" path, starting in the top-left cell and ending in one of the bottom corners, while repeatedly applying the operator on a cell in the path and on the next, adjacent one. All the cells in the path, apart from the last one will become white. There are initial states for which the last cell will become white, and initial states for which it will not. This is justified by the following *invariant property*.

> *The parity of the number of white cells (as well as back cells) never changes.*

If the initial parity of the number of white cells is different from the total number of cells in the matrix, then the goal may not be achieved.

Part-B is subtler. Since the operator operates on three cells at a time, parity of the total number of cells of a particular color is not preserved. We start by trying to whiten the matrix in a "snake-like" path, and progress slightly different in the bottom two lines. Progression in these two lines may be done concurrently, from left to right, until two cells remain – the right-bottom cell and an adjacent one. If all the matrix becomes white, then we are done. But, one or both of the last two cells may be black.

At this point we may seek an illuminating property, or pattern. The operator is applied on three adjacent cells in every application. It may be useful to *divide the matrix cells into three groups*, so that during the "snake like" process, the operator will be applied on one cell from each group. One way of doing so is by adding an auxiliary number to each cell in a "diagonal manner" as follows.

1	2	3	1	2	3	1
3	1	2	3	1	2	3
2	3	1	2	3	1	2
1	2	3	1	2	3	1

Each cell belongs to group1, group2, or group3. It is always possible to apply `xorL` on three cells, such that each is from a different group. In the above diagram, the size of group1 is 10 (cells), the size of group2 is 9 and the size of group3 is 9. It may be proved (possibly by induction), that the difference between the sizes of every two groups is at most 1. This will help us later.

If we implement the "snake like" scheme (with special progress in the two bottom lines) on the above example matrix, we will end up with an all-white matrix, accept for the cell numbered 2 in the right column (which will be black).

Why is that? When we count the initial number of black cells in each group, we notice that that there are three black 1's, four black 2's, and five black 3's. The parity of the number of black 2's is different from that of black 1's and black 3's. When `xorL` is applied, it inverts the color of one cell in each group. This implies the following *invariant property*.

> *Every application of* `xorL` *maintains the <u>difference of the parities of the number of black cells</u> between two groups.*

The above invariant implies that if there is a parity difference between the initial number of black cells of two groups, then this difference will remain; and there will never be a point in which the number of black cells in both groups will be concurrently zero. This explains the result of trying to whiten the matrix of the above example. All the cells of group1 and all the cells of group3 may be white, but one cell of group2 may remain black.

We mentioned earlier that in the numbering of cells, the differences between the amounts of 1's, 2's and 3's are at most 1. This implies that if the parities of the amounts of black cells are initially equal in the groups, then the matrix <u>can</u> be whitened. If the amounts of 1's, 2's, and 3's could be larger than 1, then the condition of equal parities would be an insufficient condition for whitening the matrix, since the amount of black cells could be 0 in one group and 2 in the other.

In Part-C, it is possible to whiten the whole matrix with the operator `Xor3`, except for a 2×2 structure of cells that will remain. In this structure, two groups (out of 1, 2, 3) will have one representative, and one group will have two representatives. While the cells of the two groups with one representative may both be white, the two cells of the third group may be both white or both black. What is a property of the initial number of black cells that guarantees that these two cells will be white in the end of the whitening process? We leave this question to the interested reader.

All in all, the key element that paved the way to the solution of part-B was the *abstract view of looking at the matrix cells as cells that belong to three interleaved groups* of similar sizes, such that `xorL` may be applied each time on one cell of each group. The resulting invariant property yielded the recognition of the cases in which the matrix may be whitened and the cases in which it may not.

3. Discussion

The abstract perspectives of *ignoring details* and *relating to particular properties of recognizable parts* are relevant in problem solving. The task solutions presented here encapsulate them in computations with repeated utilizations of operators. Two operators were used for sorting and one was used for transforming binary matrix values.

The recognizable parts in the sorting tasks were pairs of integers. Unlike various sorting schemes, the focus here was not on pair adjacencies, but on pair inversions. The central element was the *invariant property* of the parity of the number of inversions. This property was the key for understanding the operators' characteristics and their limitations.

The recognizable parts in the matrix transformation tasks were borders between matrix cells. In the first matrix task, the natural tendency is to look at cell values, which are colors of areas. But the important *recognizable parts* are borders between areas; more

specifically – borders between cells of different colors. The focus here was on the *metric property* that relates the initial number of these borders to the amount that can be decreased in a single invocation of the operator. Borders between cells inside a given rectangle were *ignored*. So were possible selections between alternative choices of lengths and locations of rectangles that kept the desired property.

In the second matrix task, the recognizable parts were adjacent cells in a matrix, which *were viewed* in a way that suited a single operator application. Two *properties* led to the solution of part-B of the task: 1. The partition of the matrix cells into three groups of near sizes, so that an application of the operator on any cell of one group may also be on representatives of the other two groups; and 2. The parity differences between the number of black cells in the groups are preserved. The layout of the black and white cells in the matrix was *ignored*. So was the number of white cells. Only the number of black cells counted.

In our experience with students, the more challenging element during problem solving is the decision of what to focus on and what to ignore, as well as the kind of properties to look for. Novices tend to focus on the explicit data in a given problem, and try to associate it with their familiar cognitive schemes. However, a primary theme in solving non-routine problems is the recognition of hidden patterns and capitalization on these patterns (Schoenfeld, 1992). Such recognition should be practiced.

One relevant practice involves problems like the tasks presented here. Although these tasks may be posed as programming problems, their asset is in focusing on abstraction perspectives and verification of recognized patterns. Programming may "bypass" the latter. One may provide a suitable programming solution without sufficient insight into the problem at hand, when only input/output outcomes are examined. An "unplugged" experience is more thorough, and elaborates on the importance of comprehension and verification. The focus is primarily on the conceptual practice. No considerations of computer implementation are involved, and one focuses on exercising exploration and recognition of hidden patterns.

Learners learn definitions, theorems, and methods, but their primary mean for progress is learning from examples and practicing examples (Sinclair *et al.*, 2011). Problem solving with examples like those presented here enhances the practice of relating abstract perspectives to concrete, explicit givens. Repeated practice of relating abstraction perspectives to the concrete develops problem solving competence and enhances one's confidence in her abilities.

We embedded the tasks displayed here in the activities of our top 30 students during the beginning of their advanced training. Although the primary focus of the training was on solving IOI-like tasks, the practice of the tasks displayed here, in-between programming tasks, widened the students' viewpoint and encouraged their verification tendencies. At first, they struggled with the more challenging tasks, and were unsure about the verification of properties; but with further practice they felt more confident, realized the relevance of such tasks, and related observations in these tasks to later programming tasks. This was particularly apparent with the more competent students.

References

Fellows, M., Witten, I., Bell, T. (2005). *Computer Science Unplugged*, LuLu Pub.

Frorer, P., Hazzan, O., Manes, M. (1997). Revealing the facets of abstraction, *International Journal of Computers in Mathematical Learning*, *2*, 217–228.

Ginat, D., Blau, Y. (2017). Multiple levels of abstraction in algorithmic problem solving, *SIGCSE'48*, ACM Press, 237–242.

Hazzan, O. (2002). Reducing abstraction level when learning computability theory concepts, *ITiCSE'02*, ACM Press, 156–160.

Horváth, G., Verhoeff, T. (2002). Finding the median under IOI conditions, *Informatics in Education*, *1*, 73–92.

Knuth, D. (1973). *The Art of Computer Programming, Vol. 3*, Addison Wesley Pub.

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.

Sinclair, N., Watson, A., Zazkis, R., Mason, J. (2011). The structuring of personal spaces, *Journal of Mathematical Behavior*, *30*, 291–303.

Wing, J. (2006). Computational thinking, *Communications of the ACM*, *49*(3), 33–35.

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

Introduction of "Honorable Mention" Award at the International Olympiad in Informatics

Mile JOVANOV, Emil STANKOV

Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University
st. Rugjer Boshkovikj 16 Skopje, Macedonia
e-mail: mile.jovanov@gmail.com, emil.stankov@gmail.com

Abstract. In 2020, there are 14 International Science Olympiads for secondary school students, which aim to gather teams from all the countries around the world, with the brightest young eligible students from each country. These Olympiads are not just a science competition but a means to care for talent in the particular scientific field. International Olympiad in Informatics (IOI) is one of the first five Olympiads that arose, after Mathematics, Physics and Chemistry, and before Biology Olympiad. Being the "summit" of the brightest students, they generously award recognitions to contestants in the form of gold, silver and bronze medals, and additionally – the so called "honorable mention" award. IOI is the only Olympiad that up until 2019 has not introduced the fourth-degree award – "honorable mention". In this paper we explore the rules of the other four scientific Olympiads in order to compare their methods of awarding contestants to the current one used by IOI, and to use that analysis for proposing a rule change that will introduce "honorable mention" category at IOI. Furthermore, a set of possible approaches are considered, and for each one, the "retroactive" impact of the rule to the results of the last five IOI issues is presented. At the end, as a conclusion, the most appropriate approach is proposed.

Keywords: science competitions, programming, International Olympiad in Informatics, medals, awards, honorable mention.

1. Introduction

The **International Science Olympiads** are a group of worldwide annual competitions in various areas of science. In 2020, there are 14 International Science Olympiads for secondary school students, which aim to gather teams from all the countries around the world, with the brightest young eligible students from each country. These Olympiads are not just a science competition but means to care for talent in the particular scientific field. The competitions are designed for the 4–6 best high school students from each participating country selected through internal National Science Olympiads. Further in this section we present more information on the first five of these Olympiads, in chronological order of their appearance.

1.1. *International Mathematical Olympiad*

The **International Mathematical Olympiad (IMO**) is an annual mathematics competition for high school students. The first IMO was held in Romania in 1959. The problems come from various areas of mathematics, which are included in math curricula in secondary schools. Finding the solutions of these problems, however, requires exceptional mathematical ability and excellent mathematical knowledge on the part of the contestants.

The country delegation to an IMO consists of up to six student competitors and (a maximum of) two leaders. Awards are determined as follows (IMO regulations, clause 5):

- Gold medal: the top 1/12 of scores receive gold medals.
- Silver medal: the next 2/12 of scores receive silver medals.
- Bronze medal: the next 3/12 of scores receive bronze medals.
- Honorable mention: any competitor who receives a perfect score of 7 on any one question, but who does not receive a medal, is awarded an honorable mention.

1.2. *International Physics Olympiad*

The **International Physics Olympiad** (**IPhO**) is an annual physics competition for high school students. The first IPhO was held in Warsaw, Poland, in 1967.

Each national delegation is made up of at most five student competitors plus two leaders, selected on a national level. The students compete as individuals and are put to hard theoretical and laboratory examinations. According to the results, the students can be awarded gold, silver or bronze medals, or an honorable mention. The minimal scores required for Olympiad medals and honorable mentions are chosen by the organizers, according to the following rules (IPhO statutes, clause 6):

- A gold medal should be awarded to 8% of the contestants (rounded to a nearest integer).
- A silver medal or better should be awarded to 25% of the contestants (rounded to a nearest integer).
- A bronze medal or better should be awarded to 50% of the contestants (rounded to a nearest integer).
- An honorable mention or better should be awarded to 67% of the contestants (rounded to a nearest integer).
- All other participants receive certificates of participation. The participant with the highest score (absolute winner) receives a special prize, in addition to a gold medal.

1.3. *International Chemistry Olympiad*

The **International Chemistry Olympiad** (**IChO**) is an annual academic competition for high school students. It is also one of the International Science Olympiads. The first IChO was held in Prague, Czechoslovakia, in 1968.

Each delegation consists of up to four students and two mentors (one of them is designated as the head of the delegation or "head mentor"). A delegation may also include a handful of guests and scientific observers. Students must be under the age of 20 and must not be enrolled as regular students in any post-secondary education institution. All participants are ranked based on their individual scores and no official team scores are given.

Rules for awarding medals at IChO (IChO regulations, clause 15):

- The best 10% to 12% of all competitors receive gold medal.
- The next 20% to 22% receive silver medal.
- The following 30% to 32% receive bronze medals.
- An honorable mention is received by non-medalists who are in the best 70 to 71% of all competitors.
- The exact number of recipients for each award is determined automatically to yield the largest possible difference in the marks of students receiving different honors. In the case of identical differences, the one resulting in more medals will be selected.
- Each medalist must receive the medal and a corresponding certificate from the organizer.
- Other prizes may be awarded in addition to the medals.
- Each competitor receives a certificate of participation.
- In the awarding ceremony, the non-medalists are called alphabetically.

1.4. *International Olympiad in Informatics*

The **International Olympiad in Informatics** (**IOI**) is an annual competitive programming competition for high school students. It is the second largest science Olympiad, after the International Mathematical Olympiad, in terms of number of participating countries (87 at IOI 2019). The first IOI was held in 1989, in Pravetz, Bulgaria.

Students at the IOI compete on an individual basis, with up to four students competing from each participating country. Students in the national teams are selected through national computing contests, and they are led by one or two team leaders from the country. The contest consists of two days of computer programming/coding and problem-solving of algorithmic nature. The knowledge and skills necessary to solve the tasks are on very high level, often compared to the content of the most competitive algorithmic courses at the universities.

The scores from the two competition days and all problems are summed up separately for each contestant. At the awarding ceremony, contestants are awarded medals depending on their relative total score (IOI regulations, clause S6.11). No more than half of the contestants are to receive medals on the basis that:

- About one twelfth of all contestants receive a gold medal.
- About one sixth of all contestants receive a silver medal.
- About one quarter of all contestants receive a bronze medal.

More exact algorithm is given in the E6.11 of the IOI regulations, which states: Medal boundaries are allocated by the following rules:

- The score necessary to achieve a gold medal is the largest score such that at least one twelfth of all contestants receive a gold medal.
- The score necessary to achieve a silver medal is the largest score such that at least one quarter of all contestants receive a gold or silver medal.
- The score necessary to achieve a bronze medal is the smallest score such that at most one half of all contestants receive a medal.

1.5. *International Biology Olympiad*

The **International Biology Olympiad** (**IBO**) is an annual science Olympiad for high school students under the age of 20. All participating countries send the four winners of their National Biology Olympiad to the IBO, usually accompanied by two adults who are members of the international jury for the duration of the competition. The first IBO was held in Czechoslovakia in 1990, with 6 participating countries. Nowadays, there are up to 78 participating countries (at IBO 2019).

The awards are determined according to the cutoffs below, where n is the number of competitors, and $\lceil n \rceil$ is the ceiling function (e.g. $\lceil 4.1 \rceil = 5$, $\lceil 4.9 \rceil = 5$). The maximum number of awards equals $0.7 \lceil n \rceil + 2$ (IBO guidelines).

Gold medal	$w = \lceil 0.1\, n \rceil$	The last gold medal winner is the one preceding the largest gap out of the three following the top w competitors
Silver medal	$x = \lceil 0.3\, n \rceil$	The last silver medal winner is the one preceding the largest gap out of the three following the top x competitors
Bronze medal	$y = \lceil 0.6\, n \rceil$	The last bronze medal winner is the one preceding the largest gap out of the three following the top y competitors
Certificate of merit	$z = \lceil 0.7\, n \rceil$	The last certificate of merit winner is the one preceding the largest gap out of the three following the top z competitors

In the following section we will provide statistics regarding the number of medals awarded at the year 2019's issues of the Physics, Chemistry and Biology Olympiads, since they all have "steady" principles for awarding medals and Honorable mentions/

Certificates of merit. We will look into the results from IMO more deeply, considering more of the last issues, since there is a specific condition for awarding the Honorable mention awards. Further in the paper we will list several different proposals/approaches for introducing the Honorable mention at IOI, and for each one, the "retroactive" impact of the approach to the results of the last five IOI issues will be analyzed. At the end, as a conclusion, based on the conducted analysis, the most appropriate approach will be proposed.

2. Awards Presented at Other Science Olympiads

Science Olympiads are not just a science competition but a means to care for talent in the particular scientific field. International Olympiad in Informatics (IOI) is one of the first five Olympiads that arose, after Mathematics, Physics and Chemistry, and before Biology Olympiad. Being the "summit" of the brightest students, at all Olympiads contestants are generously awarded with recognitions in the form of gold, silver and bronze medals, and additionally, the so called "Honorable mention" award.

Here we provide statistics regarding the number of medals gained at the year 2019's issues of the Physics, Chemistry and Biology Olympiads, since they all have "steady" principles for awarding medals and Honorable mentions/Certificates of merit.

According to their official statistics sites / official published documents, in Table 1, Table 2 and Table 3 we may see the number of awards presented at each Olympiad.

Table 1

Awards presented at IPhO 2019

Total # of contestants 363 from 78 countries	#	%	Cumulative %
Gold medal	34	9.37%	
Silver medal	66	18.18%	
Bronze medal	101	27.82%	55.37%
Honorable mention	50	13.77%	69.15%

Table 2

Awards presented at IChO 2019

Total # of contestants 309 from 80 countries	#	%	Cumulative %
Gold medal	37	12.33%	
Silver medal	64	21.33%	
Bronze medal	95	31.67%	63.43%
Honorable mention	23	7.67%	71.10%

Table 3

Awards presented at IBO 2019

Total # of contestants 285	#	%	Cumulative %
Gold medal	31	10.88%	
Silver medal	55	19.30%	
Bronze medal	87	30.53%	60.70%
Honorable mention	27	9.47%	70.18%

As a general conclusion, all three Olympiads award more than 50% medals (55.4% – 63.4%) which is more generous than the IOI rule of awarding no more than 50% of the contestants with medal. Even more, if we include the contestants awarded with Honorable mention (or Certificate of Merit at IBO), we may see that the total number of awards is around 70% of all participants (69.2% – 71.1%). Since IOI did not award Honorable mention award until IOI 2019, obviously the percentage of awards given at IOI is 20 percent points smaller than other Olympiads, or in percentages, 29% less awards.

2.1. *Awards at IMO in the Last 4 Years*

We will look into the results from IMO more deeply, considering more of the last issues, since there is a specific condition for awarding the Honorable mention awards.

Table 4 shows the number of awarded contestants from IMO in the years 2016 to 2019, according to IMO statistics (IMO 2017). At IMO, any competitor who receives a perfect score of 7 on any one question, but who does not receive a medal, is awarded

Table 4

Awarded contestants at IMO in the years 2016–2019

	2016 year		2017 year		2018 year		2019 year	
	Number of awarded contestants	Awarded contestants (%)	Number of awarded contestants	Awarded contestants (%)	Number of awarded contestants	Awarded contestants (%)	Number of awarded contestants	Awarded contestants (%)
Gold medal	44	7.31	48	7.80	48	8.08	52	8.37
Silver medal	101	16.78	90	14.63	98	16.50	94	15.14
Bronze medal	135	22.43	153	24.88	143	24.07	156	25.12
Honorable mention	162	26.91	222	36.10	138	23.23	144	23.19
Total awards	442	73.43	513	83.41	427	71.88	446	71.82
Total participants	602		615		594		621	

an honorable mention. From the table we can see that the number of students who are awarded with honorable mention is approximately 23% – 37%. In total, IMO awards almost same percentage of medals as IOI, but more total awards (71.8% – 83.4%) compared to 50% at IOI.

As a conclusion, it is obvious that IOI should introduce an additional, fourth level award, in order to equalize the possibility of wining an award for its contestants, compared to other Olympiads.

3. Different Approaches for Introduction of Honorable Mention at IOI

During the IOI 2019, on the initiative of the authors of this paper, a discussion on the introduction of Honorable mention (HM) award was held for the interested team leaders. Different approaches for awarding this fourth level award were proposed and discussed. In this paper we are analyzing five different approaches/scenarios that are based mainly on the experiences from the other science Olympiads, as well as ideas from the leaders based on the experiences of regional Olympiads like the Balkan Olympiad in Informatics, for example.

For every approach, we provide different tables and accompanying graphic that show the important characteristics of the approach, such as:

1. Number and percentage of awarded contestants, that will show how many contestants gain additionally award, on top of the ones awarded with medals.
2. Range of points that qualify the contestant to score a medal or HM, in order to see if some approach awards HM for some students with rather small number of points, for example, or average points for the contestants that gain a particular award.
3. Number of countries that haven't won a medal for the particular year, but would have won one or more HM awards, etc.

The data in the tables that follow originate from the last five International Olympiads in Informatics (years 2015–2019), according to the IOI statistics pages (IOI 2015, IOI 2016, IOI 2017, IOI 2018, IOI 2019). We consider a hypothetical situation in which we apply the proposed approach for HM over the already scored points of the contestants. Analyzed scenarios are:

*   Honorable mention for the following 15% of contestants who did not win a medal.
*   Honorable mention for contestants who have correctly solved at least one task (received 100 points), but did not win a medal.
*   Honorable mention for contestants who have won at least 50% of the points won by the last bronze medalist.
*   Honorable mention for the following 20% of contestants who did not win a medal.
*   Honorable mention for the following 20% of contestants who did not win a medal but have also won at least 50% of the points won by the last bronze medalist.

3.1. *Honorable Mention for the Following 15% of Contestants Who did not Win a Medal*

The data in Table 5, Table 6 and Fig. 1 show that if this approach is used, then all contestants awarded with HM will have rather close scores among each other, i.e. the contestant who will receive the last HM will still have a significant number of points, compared to the one with HM who is the first non-medalist.
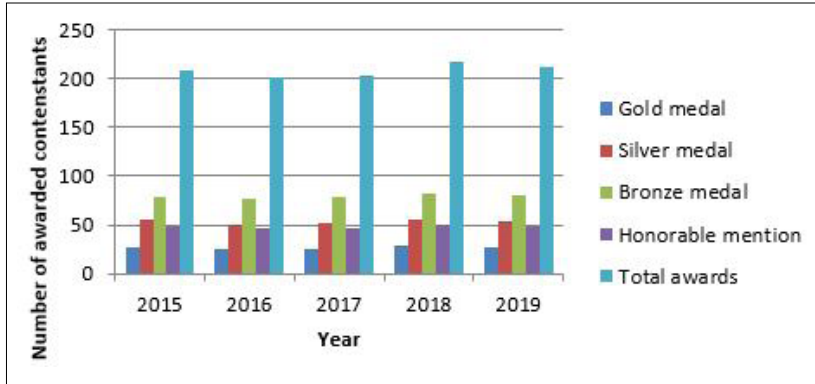
Fig. 1. Awarded contestants at IOI (HM 15%).

Table 5

Honorable mention for the following 15% of contestants who did not win a medal
(GM = Gold medal, SM = Silver m., BM = Bronze m., HM = Honorable mention,
TA = Total awards, TP = Total partic.)

| | 2015 year | | | 2016 year | | | 2017 year | | | 2018 year | | | 2019 year | | |
	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)
GM	27	8.4	73.4 - 100	26	8.4	69.3 - 99.5	26	8.4	58.8 - 98.2	29	8.7	56.0 - 83.2	28	8.6	69.1 - 91.2
SM	55	17.1	54.3 - 72.9	51	16.6	54.7 - 68.8	52	16.9	41.6 - 58.5	55	16.4	45.3 - 55.7	54	16.6	54.9 - 68.6
BM	79	24.5	30.9 - 53.9	77	25.0	40.0 - 53.3	78	25.3	22.9 - 41.5	83	24.8	31.2 - 45.2	81	24.8	41.7 - 54.8
HM	48	14.9	20.7 - 30.8	47	15.3	30.2 - 39.8	46	14.9	16.5 - 22.6	50	14.9	24.3 - 30.8	49	15.0	34.2 - 41.6
TA	209	64.9		201	65.3		202	65.6		217	64.8		212	65.0	
TP	322			308			308			335			326		

Table 6

Honorable mention for the following 15% of contestants who did not win a medal – average points

	2015 year		2016 year		2017 year		2018 year		2019 year	
	Points (%)	Average points (%)	Points (%)	Average points (%)	Points (%)	Average points (%)	Points (%)	Average points (%)	Points (%)	Average points (%)
Gold medal	73.40 - 100.00	86.7	69.33 - 99.50	84.42	58.84 - 98.25	78.55	56.00 - 83.17	69.59	69.12 - 91.18	80.15
Silver medal	54.26 - 72.92	63.59	54.67 - 68.83	61.75	41.57 - 58.50	50.34	45.33 - 55.67	50.5	54.86 - 68.58	61.72
Bronze medal	30.91 - 53.92	42.42	40.00 - 53.33	46.67	22.86 - 41.50	32.18	31.17 - 45.17	38.17	41.70 - 54.77	48.24
Honorable mention	20.67 - 30.85	25.76	30.17 - 39.83	35.00	16.48 - 22.63	19.56	24.33 - 30.83	27.58	34.18 - 41.60	37.89
Mean		54.62		56.96		45.16		46.46		57.00
Deviation		22.87		18.47		22.17		15.62		15.81

In Table 7, we analyze the number of countries that would have been included in the "awarded" countries, i.e., countries that don't have a contestant who has won a medal, but would have had contestant with HM.

With this approach additional 11 countries in each year would have entered in the group of "awarded" countries. Table 8 further shows that using this approach we decrease the number of "non-awarded" countries from approx. 26 to approx. 15.

Table 7

Analysis of the number of gained honorable mentions and medals with honorable mentions by countries, if honorable mention is given to the following 15% of contestants who did not win a medal (HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of countries that won only HM	7	11	9	4	6
Number of countries that won more than one HM	4	/	2	7	5
Number of countries that won a medal and one or more HM	6	12	12	9	8
Number of countries that won more than one medal and one or more HM	15	18	11	16	12
Total	32	41	34	36	31

Table 8

Analysis of the number of countries that did not receive a medal but would receive
an honorable mention if an honorable mention is given to the following 15% of
contestants who did not win a medal (HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of countries that won one HM	7	11	9	4	6
Number of countries that won more than one HM	4	/	2	7	5
Number of countries that did not win a HM	15	16	12	15	17
Total	25	27	23	26	28

3.2. *Honorable Mention for Contestants Who Have Correctly Solved at Least One Task (Received 100 Points), but did not Win a Medal*

The data in Table 9 show that if this approach is used, there would be great discrepancies among the number of participants that would win HM in each year. We furthermore explore additional characteristics in Table 10.

Table 10 shows the number of contestants who would win an honorable mention, their average points in %, lowest points of contestant who would win an honorable mention, the percentile of that contestant, and the number of contestants that would have a higher score than the lowest ranked contestant with honorable mention, but would not win an honorable mention.

Table 9

Honorable mention for contestants who have correctly solved at least one task (received
100 points), but did not win a medal (GM = Gold medal, SM = Silver medal, BM = Bronze
medal, HM = Honorable mention, TA = Total awards, TP = Total participants)

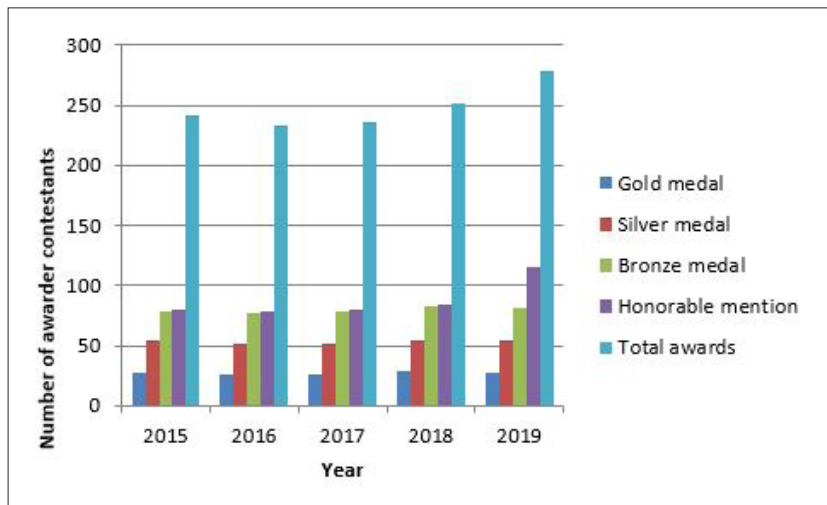|	2015 year			2016 year			2017 year			2018 year			2019 year		
	Number of awarded contestants	Awarded contestants (%)	Average points (%)	Number of awarded contestants	Awarded contestants (%)	Average points (%)	Number of awarded contestants	Awarded contestants (%)	Average points (%)	Number of awarded contestants	Awarded contestants (%)	Average points (%)	Number of awarded contestants	Awarded contestants (%)	Average points (%)
GM	27	8.4	86.7	26	8.4	84.4	26	8.4	78.6	29	8.7	69.6	28	8.6	80.1
SM	55	17.1	63.6	51	16.6	61.7	52	16.9	50.0	55	16.4	50.5	54	16.6	61.7
BM	79	24.5	42.4	77	25.0	46.7	78	25.3	32.2	83	24.8	38.2	81	24.8	48.2
HM	**5**	1.6	23.2	**31**	10.1	32.6	**3**	1.0	20.4	**47**	14.0	24.5	**37**	11.3	35.4
TA	166	51.6		185	60.1		159	51.6		214	63.9		200	61.3	
TP	322			308			308			335			326		

Table 10

Honorable mention for contestants who have correctly solved at least one task
(received 100 points), but did not win a medal – additional characteristics
(HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of awarded contestants with HM	5	31	3	47	37
Average Points (%)	23.17	32.57	20.42	24.54	35.38
Lowest points	117	115	113	102	130.39
Percentile	19.50	19.17	18.83	17.00	21.73
Number of contestants who scored higher than the lowest ranked HM and did not win HM	55	52	25	34	75

The provided data present many negative characteristics of this approach. For example: unpredictable number of contestants that gain HM, usually rather small number of them; rather low total points for some contestants that will gain a HM award; rather big number of participants that will have more points than someone with HM, but won't get an award. Thus, this approach, based on the IMO approach for HM awarding, is not suitable for the Informatics Olympiad. We may further analyze the results of the Olympiad from year 2019. There are 37 students who have correctly solved at least one task (received 100 points) and haven't gained any medal. However, there is a big difference in the points and places in the scoreboard. For example, the best of these participants has a total of 249.58 points or 41.60%, while the last participant with at least one correctly solved task has 130.39 points or 21.73%, which represents a big difference in the points and places of the awarded participants.

3.3. *Honorable Mention for Contestants Who Have Won at Least 50% of the Points Won by the Last Bronze Medalist*

Another interesting approach is to award with HM the contestants who have won at least 50% of the points won by the last bronze medalist. This approach guaranties that every contestant that will win HM, will still have a significant number of points – not less than half of the points scored by the last bronze medalist. With this approach, the number of HM winners fluctuates, but, as seen in Table 11 and Fig. 2, is rather stabilized around 25% of the total number of participants, with the exception in 2019. Table 12 presents some additional characteristics for the approach.

In Table 13 and Table 14, we analyze the number of countries that would have been included in the "awarded" countries, that is, countries that don't have a contestant that has won a medal, but would have had contestant with HM. In Table 14 we may see that with this approach we decrease the number of "non-awarded" countries to only 8 in years 2015 and 2019, and up to only 11 in 2016.

Table 11

Honorable mention for contestants who have won at least 50% of the points won by the last bronze medalist (GM = Gold medal, SM = Silver medal, BM = Bronze medal, HM = Honorable mention, TA = Total awards, TP = Total participants)

	2015 year			2016 year			2017 year			2018 year			2019 year		
	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)
GM	27	8.4	73.4 - 100	26	8.4	69.3 - 99.5	26	8.4	58.8 - 98.2	29	8.7	56.0 - 83.2	28	8.6	69.1 - 91.2
SM	55	17.1	54.3 - 72.9	51	16.6	54.7 - 68.8	52	16.9	41.6 - 58.5	55	16.4	45.3 - 55.7	54	16.6	54.9 - 68.6
BM	79	24.5	30.9 - 53.9	77	25.0	40.0 - 53.3	78	25.3	22.9 - 41.5	83	24.8	31.2 - 45.2	81	24.8	41.7 - 54.8
HM	80	24.8	15.5 - 30.8	79	25.6	21.2 - 39.8	80	26.0	11.5 - 22.6	84	25.1	16.2 - 30.8	115	35.3	20.9 - 41.6
TA	241	74.8		233	75.6		236	76.6		251	74.9		278	85.3	
TP	322			308			308			335			326		

Fig. 2. Awarded contestants at IOI (HM for at least 50% of the points won by the last bronze medalist).

Table 12

Honorable mention for contestants who have won at least 50% of the points won by the last bronze medalist – additional characteristics

	2015 year		2016 year		2017 year		2018 year		2019 year	
	Points (%)	Average points (%)	Points (%)	Average points (%)	Points (%)	Average points (%)	Points (%)	Average points (%)	Points (%)	Average points (%)
Gold medal	73.40 - 100.00	86.7	69.33 - 99.50	84.42	58.84 - 98.25	78.55	56.00 - 83.17	69.59	69.12 - 91.18	80.15
Silver medal	54.26 - 72.92	63.59	54.67 - 68.83	61.75	41.57 - 58.50	50.34	45.33 - 55.67	50.5	54.86 - 68.58	61.72
Bronze medal	30.91 - 53.92	42.42	40.00 - 53.33	46.67	22.86 - 41.50	32.18	31.17 - 45.17	38.17	41.70 - 54.77	48.24
Honorable mention	15.46 - 30.85	23.16	21.17 - 39.83	30.50	11.46 - 22.63	17.05	16.17 - 30.83	28.08	20.91 - 41.60	31.26
Mean		53.97		55.84		44.53		46.59		55.34
Deviation		23.70		19.86		22.91		15.47		17.93

Table 13

Analysis of the number of gained honorable mentions and medals with honorable mentions by countries, if HM is given to contestants who have won at least 50% of the points won by the last bronze medalist (HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of countries that won only HM	10	11	6	6	5
Number of countries that won more than one HM	7	5	8	10	17
Number of countries that won a medal and one or more HM	9	13	14	13	11
Number of countries that won more than one medal and one or more HM	17	24	17	22	25
Total	43	53	45	51	58

Table 14

Analysis of the number of countries that did not receive a medal but would receive an honorable mention, if HM is given to contestants who have won at least 50% of the points won by the last bronze medalist (HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of countries that won one HM	10	11	6	6	5
Number of countries that won more than one HM	7	5	8	10	15
Number of countries that did not win a HM	8	11	9	10	8
Total	25	27	23	26	28

3.4. *Honorable Mention for the Following 20% of Contestants Who did not Win a Medal*

This approach is in the same spirit as the one analyzed in section 3.1 (HM for following 15% of the contestants), but with a different percentage. The corresponding data are given in Table 15, Table 16 and Table 17.

This percentage leads to rather similar number of awards, compared to other Scientific Olympiads (analyzed in Section 2), and as seen in Table 17, it manages to decrease the number of "non-awarded" countries from approx. 26 to only 12 to 14 countries.

Table 15

Honorable mention for the following 20% of contestants who did not win a medal
(GM = Gold medal, SM = Silver medal, BM = Bronze medal, HM = Honorable mention,
TA = Total awards, TP = Total participants)

	2015 year			2016 year			2017 year			2018 year			2019 year		
	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)	Number of awarded contestants	Awarded contestants (%)	Points (%)
GM	27	8.4	73.4 - 100	26	8.4	69.3 - 99.5	26	8.4	58.8 - 98.2	29	8.7	56.0 - 83.2	28	8.6	69.1 - 91.2
SM	55	17.1	54.3 - 72.9	51	16.6	54.7 - 68.8	52	16.9	41.6 - 58.5	55	16.4	45.3 - 55.7	54	16.6	54.9 - 68.6
BM	79	24.5	30.9 - 53.9	77	25.0	40.0 - 53.3	78	25.3	22.9 - 41.5	83	24.8	31.2 - 45.2	81	24.8	41.7 - 54.8
HM	64	19.9	18.6 - 30.8	64	20.8	25.7 - 39.8	62	20.1	13.7 - 22.6	67	20.0	19.0 - 30.8	66	20.2	32.0 - 41.6
TA	225	69.9		216	70.8		218	70.8		234	69.9		229	70.2	
TP	322			308			308			335			326		

Table 16

Analysis of the number of gained honorable mentions and medals with honorable mentions by countries, if HM is given to the following 20% of contestants who did not win a medal (HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of countries that won only HM	7	12	7	3	7
Number of countries that won more than one HM	6	1	4	9	7
Number of countries that won a medal and one or more HM	7	14	12	10	10
Number of countries that won more than one medal and one or more HM	17	19	15	18	19
Total	37	46	38	40	43

Table 17

Analysis of the number of countries that did not receive a medal but would receive an honorable mention if an honorable mention is given to the following 20% of contestants who did not win a medal (HM = Honorable mention)

	2015 year	2016 year	2017 year	2018 year	2019 year
Number of countries that won one HM	7	12	7	3	7
Number of countries that won more than one HM	6	1	4	9	7
Number of countries that did not win a HM	12	14	12	14	14
Total	25	27	23	26	28

3.5. *Honorable Mention for the Following 20% of Contestants Who did not Win a Medal but also Have Won at Least 50% of the Points Won by the Last Bronze Medalist*

This approach builds on the idea to award Honorable mention for the following 20% of contestants who will not win a medal, but with a second condition that the contestants must win at least 50% of the points won by the last bronze medalist. The approach guaranties that every contestant that will win HM, will still have a significant number of points – not less than half of the points scored by the last bronze medalist, and that only up to 70% (50+20) of the total contestants will win an award. In theory, this is a better approach than the clear one with only 20% additional contestants, but if we look at Table 15, we will see that in each of the last 5 years, every contestant in the range satisfies this additional condition. This is not a guarantee that it will hold always in the future, but an irregularity is highly unexpected.

4. Conclusion

In this paper we explored the possibility of introducing a fourth level award for the IOI contestants called Honorable mention. Firstly, we presented the rules for awarding recognitions of other four scientific Olympiads (in Mathematics, Physics, Chemistry and Biology) as the oldest five Olympiads including the IOI. Then, we presented data from the last issues of IMO, IPhO, IChO and IBO, in order to show how they implement their rules in practice. After that, we analyzed five different approaches for introducing Honorable Mention at IOI. The results of the analysis were given in series of Tables, as well as Figures. Main results are summarized in Table 18. According to the analysis, the best approach, most similar to other Olympiads, with most clear wording is the one analyzed in the subsection 3.4., i.e., Honorable mention to be awarded to the following 20% of contestants who did not win a medal, getting to the cumulative 70% of awarded contestants.

Based on the data in this paper IC decided to propose to GA introduction of HM at IOI according to the above rule. We believe that this paper will give the rationale behind that decision of the IOI community.

Table 18

Number of awarded contestants with honorable mention from all analyzed scenarios (HM 15% = Honorable mention for 15%, HM 1 = Honorable mention for correctly solved at least one task, HM 50% = Honorable mention for at least 50% of the points won by the last bronze medalist, HM 20% – Honorable mention for 20%, BM = Bronze medalist, TP = Total participants)

	2015 year				2016 year				2017 year				2018 year				2019 year			
	Number of awarded contestants	Awarded contestants (%)	Points (%)	Points of the last HM winner vs points of the last BM	Number of awarded contestants	Awarded contestants (%)	Points (%)	Points of the last HM winner vs points of the last BM	Number of awarded contestants	Awarded contestants (%)	Points (%)	Points of the last HM winner vs points of the last BM	Number of awarded contestants	Awarded contestants (%)	Points (%)	Points of the last HM winner vs points of the last BM	Number of awarded contestants	Awarded contestants (%)	Points (%)	Points of the last HM winner vs points of the last BM
HM 15%	48	14,90	20,67 - 30,85	20,67 / 30,91	47	15,26	30,17 - 39,83	30,17 / 40,00	46	14,94	16,48 - 22,63	16,48 / 22,86	50	14,93	24,33 - 30,83	24,33 / 31,17	49	15,03	34,18 - 41,60	34,18 / 41,70
HM 50%	80	24,84	15,46 - 30,85	15,46 / 30,91	79	25,65	21,17 - 39,83	21,17 / 40,00	80	25,97	11,46 - 22,63	11,46 / 22,86	84	25,07	16,17 - 30,83	13,17 / 31,17	115	35,28	20,91 - 41,60	20,91 / 41,70
HM 1	5	1,55	23,17	19,50 / 30,91	31	10,06	32,57	19,17 / 40,00	3	0,97	20,42	18,83 / 22,86	47	14,03	24,54	17,00 / 31,17	37	11,35	35,38	21,73 / 41,70
HM 20%	64	19,86	18,58 - 30,85	18,58 / 30,91	64	20,78	25,67 - 39,83	25,67 / 40,00	62	20,13	13,70 - 22,63	13,70 / 22,86	67	20	19,00 - 30,83	19,00 / 31,17	66	20,25	32,00 - 41,60	32,00 / 41,70
TP	322				308				308				335				326			

Acknowledgement

References

IBO – International Biology Olympiad (1990–2020).
  `https://www.ibo-info.org` (accessed 26/7/2020).
IBO guidelines – Operational Guidelines of the International Biology Olympiad.
  `https://www.ibo-info.org/en/info/rules-guidelines.html` (accessed 26/7/2020).
IChO – International Chemistry Olympiad (1968–2020).
  `http://www.ichosc.org` (accessed 26/7/2020).
IChO regulations – Regulations of the International Chemistry Olympiad.
  `http://www.ichosc.org/regulations` (accessed 26/7/2020).
IMO – International Mathematical Olympiad (1959–2020).
  `http://www.imo-official.org` (accessed 26/7/2020).
IMO regulations – Regulations of the International Mathematical Olympiad.
  `http://www.imo-official.org/documents/RegulationsIMO.pdf` (accessed 26/7/2020).
IMO 2017 – Statistics page of the International Mathematical Olympiad 2017.
  `http://www.imo-official.org/year_statistics.aspx?year=2017` (accessed 26/7/2020).
IOI – International Olympiad in Informatics (1989–2020).
  `http://ioinformatics.org` (accessed 26/7/2020).
IOI regulations – Regulations of the International Olympiad in Informatics.
  `http://ioinformatics.org/files/regulations19.pdf` (accessed 26/7/2020).
IOI 2015 – Statistics page of the International Olympiad in Informatics 2015.
  `http://stats.ioinformatics.org/results/2015` (accessed 26/7/2020).
IOI 2016 – Statistics page of the International Olympiad in Informatics 2016.
  `http://stats.ioinformatics.org/results/2016` (accessed 26/7/2020).
IOI 2017 – Statistics page of the International Olympiad in Informatics 2017.
  `http://stats.ioinformatics.org/results/2017` (accessed 26/7/2020).
IOI 2018 – Statistics page of the International Olympiad in Informatics 2018.
  `http://stats.ioinformatics.org/results/2018` (accessed 26/7/2020).
IOI 2019 – Statistics page of the International Olympiad in Informatics 2019.
  `http://stats.ioinformatics.org/results/2019` (accessed 26/7/2020).
IPhO – International Physics Olympiad (1967–2020).
  `http://www.ipho-new.org` (accessed 26/7/2020).
IPhO statutes – Statutes of the International Physics Olympiad.
  `http://www.ipho-new.org/statutes-syllabus` (accessed 26/7/2020).

M. Jovanov is an associate 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.

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.

CSES – Yet Another Online Judge

Antti LAAKSONEN, Topi TALVITIE
University of Helsinki, Department of Computer Science
e-mail: antti.laaksonen@helsinki.fi, topi.talvitie@helsinki.fi

Abstract. This paper describes the current state and future plans of the Code Submission Evaluation System (CSES) online judge project. Since 2013, CSES has been used to organize several online programming courses and contests in Finland, including algorithm courses at the University of Helsinki, the yearly Finnish Olympiad in Informatics (Datatähti), and the Baltic Olympiad in Informatics 2016. CSES is also known for the CSES Problem Set project whose purpose is to create a high quality problem collection for learning algorithmic problem solving, and also to document the history of programming problems.

Keywords: online judge, problem set, problem solving, competitive programming.

1. Introduction

The Code Submission Evaluation System (CSES) project started in 2013, and an early version of the system was used in a competitive programming camp organized at the University of Helsinki in summer 2013. The initial goal of the project was to establish a small system that could be used in local contests and training – in fact, many parts of the original system were implemented just the night before the first day of the camp.

In 2015, CSES was re-implemented from scratch and the domain cses.fi was registered. The new system was first used to organize an algorithm programming course at the University of Helsinki and the final round of the Finnish Olympiad in Informatics. However, the real ordeal for CSES occurred a year later when the Baltic Olympiad in Informatics 2016 was hosted by Finland and CSES was used to organize the contest.

During the first years, CSES was only used in Finland, but the situation changed in 2017 when the first version of the CSES Problem Set problem collection was published. Today CSES is an international system and most users come outside Finland. At the time of writing, the ten most active countries (in terms of the number of submissions) are Finland, India, United States, Russia, Vietnam, Croatia, Brazil, Morocco, Argentina and Bangladesh.

Recently, after releasing an updated version of the problem set, CSES has become a quite popular system in the competitive programming world. At the time of writing, the total number of submissions is about 700,000 and there are 5,000 new submissions every day.

2. System Overview

2.1. *Features and Design*

The two main features of CSES are courses and contests. A course consists of programming problems with automatic evaluation and text pages that can be used as online learning material. The most important course in the system is the CSES Problem Set; Fig. 1 shows an example problem statement in the course.

CSES supports both IOI style contests, where problems can have subtasks and partial scores, and ICPC style contests, where a submission is either accepted or not. CSES is

Fig. 1. A problem in the CSES Problem Set.

Fig. 2. An IOI style scoreboard in the Datatähti Open 2019 contest.

used to organize the Finnish Olympiad in Informatics (Datatähti), and in recent years there have also been international versions of final rounds in the Datatähti Open contest series. As an example, Fig. 2 shows the scoreboard of Datatähti Open 2019.

While most problems in CSES are standard input/output tasks, CSES also supports other problem types, such as output only tasks, interactive tasks, library tasks and encoder/decoder tasks. CSES can also be used to organize virtual contests where each contestant has an individual time window. At the moment, CSES provides two virtual contest archives that contain all BOI and CEOI problems from 2005–2019.

Another feature, used in the CSES Problem Set, is hacking: after solving a problem, it is possible to view submissions from other users and try to find a test that breaks them. Fig. 3 shows the interface used in hacking. After a successful hacking attempt, the new test is automatically added to the test data and all submissions to the problem are re-evaluated.

To make sure that problems have correct test data and can be hacked, each problem has a validator that automatically checks the input format. In IOI style contests, validators are also used to determine which subtasks a test input belongs to.

Fig. 3. In hacking, you can view other people's submissions
and try to send a new test input that breaks the code.

2.2. *Technical Details*

CSES consists of two separate parts: the website and the judge servers. The website is implemented using PHP and PostgreSQL. The submissions are evaluated by a distributed cluster of judge servers. The judging environment is implemented in Rust, and to run the user-supplied code in a sandbox where it cannot harm the rest of the system, it uses the Isolate sandboxing utility (Mare , Blackham, 2012), which in turn uses the namespace and control group features of the Linux kernel.

To avoid timing interference between jobs, each judge server runs at most one job at a time. Because of this, a large number of servers is required, however they need not to be powerful in the parallel processing sense. The Intel NUC7i3 Mini-PC was found to be a cost-effective choice of hardware for this use case. To keep the cost down, the servers do not have internal mass storage drives; instead, the root filesystem is loaded into memory from a USB stick. The servers are upgraded by updating the filesystem images over the network.

3. CSES Problem Set

The CSES Problem Set is an ongoing project whose purpose is to create a high quality collection of educational algorithm programming problems. At the time of writing the problem set consists of 200 problems, and the final goal is to reach 1,000 problems.

Many problems in the problem set are *introductory problems* that teach problem solving techniques, such as how to use a certain data structure or how to apply an algorithm design idea. Here are examples of such problems:

- Given a set of coins, count the number of distinct ways you can create a sum when you can use each coin any number of times. (topic: dynamic programming).
- Create a data structure that efficiently supports the following operations: (1) update an array value, (2) calculate the sum of values in a range. (topic: segment tree).
- Count the number of permutations of 1, 2, …, *n* where each position *i* has some other value than *i*. (topic: combinatorics).

In addition, there are *advanced problems* that are more difficult and require more thinking. For example, such problems are:

- Given a grid whose each square is either black or white, efficiently find the largest rectangle where each square is white.
- Given a directed graph, add the minimum number of edges after which the graph is strongly connected.

There are some challenges in creating a large problem set: how to ensure the quality of the problems, how to classify the problems, and how to even keep track of which problems have already been added to the problem set. So far, we have mainly added problems that have been used in our courses and training camps over the years, but we expect that new tools will be needed for managing the problems in the future.

Another goal in the project is to document the history of programming problems: what were the first occasions when a technique was used, how it has evolved since then and how it is used now. For example, in the context of competitive programming, some techniques have become popular after appearing at an IOI, such as the centroid decomposition technique in 2011 and the two-dimensional segment tree in 2013.

4. Other Systems

There are a large number of online judge systems available that have various features and goals (Wasik, Antczak, Badura, Laskowski, Sternal, 2018). In this section we discuss other systems that have influenced our project.

The USACO training system (Kolstad, Piele, 2007) consists of programming problems and text pages, and it can be seen as a model for the CSES course feature. An important difference is that in USACO training, only a small number of problems is available at a time and you have to solve them all to proceed, while in CSES you can

solve any problems. The USACO approach clearly has pedagogical benefits, but it also restricts the use of the problems.

The UVa online judge (Revilla, Manzoor, Liu, 2008) is a pioneering system that provides a large number of problems from ICPC contests and other sources. The problems are divided into volumes and can be solved in any order. The UVa online judge problems can be used when reading the book *Programming Challenges* (Skiena, Revilla, 2003); in a similar way, the CSES Problem Set can be used when reading the book *Guide to Competitive Programming* (Laaksonen, 2017).

The CMS system (Maggiolo, Mascellani, 2012) has been used to organize recent IOI contests, and the Kattis system (Enström, Kreitz, Niemelä, Söderman, Kann, 2011) has been used to organize the ICPC World Finals and some regional contests. The goal in the development of CSES has been to support the features of those systems (problem types and scoring) and also to implement authentic scoreboards, so that both IOI and ICPC style contests can be practiced as virtual contests.

Codeforces (2020) is a popular competitive programming platform that features weekly programming contests and a problem set that consists of all problems used in contests. The hacking feature of CSES resembles that used in Codeforces contests. Another online judge that provides problem archives from IOI style contests is oj.uz (Problems, 2020). At the moment, it contains more problems than CSES; however, not all problem types are supported.

Why are there so many online judges? A natural reason is that everyone wants to have full control of their systems and make sure they can fix them and add new features when needed. Thus, it does not seem probable that the situation would change. Still, some more collaboration would be beneficial for the competitive programming community: at the moment it is difficult to move content from one system to another, because there are no common practices on how to represent problem statements, test cases, validators, etc.

5. Acknowledgements

6. References

Codeforces (2020). Access 2020-04. `https://codeforces.com/`

Enström, E., Kreitz, G., Niemelä, F., Söderman, P., Kann, V. (2011). Five years with Kattis – using an automated assessment system in teaching. *IEEE Frontiers in Education Conference 2011*.

Kolstad, R., Piele, D. (2007). USA Computing Olympiad (USACO). *Olympiads in Informatics*, 1, 105–111.

Laaksonen, A. (2017). *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests*. Springer.

Maggiolo, S., Mascellani, G. (2012). Introducing CMS: A contest management system. *Olympiads in Informatics*, 6, 86–99.

Mare, M., Blackham, B. (2012). A new contest sandbox. *Olympiads in Informatics*, 6, 100–109.

Problems (2020). Access 2020-04. `https://oj.uz/`

Revilla, M.A., Manzoor, S., Liu, R. (2008). Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics*, 2, 131–148.

Skiena, S.S., Revilla, M.A. (2003). *Programming Challenges: The Programming Contest Training Manual.* Springer

Wasik, S., Antczak, M., Badura, J., Laskowski, A., Sternal, T. (2018). A survey on online judge systems and their applications. *ACM Computing Surveys*, 51(1), Article 3, 34 pages.
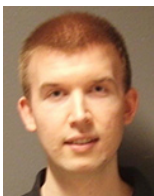
A. Laaksonen works as a university lecturer at the Department of Computer Science of the University of Helsinki. He is one of the organizers of the Finnish Olympiad in Informatics and has written a book on competitive programming. His role in the CSES project has been to implement the website and lead the creation of the CSES Problem Set.
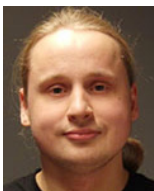
T. Talvitie works as a postdoctoral researcher at the Department of Computer Science of the University of Helsinki. His main research interest is in optimizing algorithms to run efficiently on modern hardware. In the CSES project, his role is the implementation of the judging system and the

Informatical Thinking

Michael LODI

*Department of Computer Science and Engineering, INRIA Focus, Lab. CINI Informatica e Scuola
Alma Mater Studiorum – Università di Bologna, Bologna, Italy
e-mail: michael.lodi@unibo.it*

Abstract. In this paper, we reviewed many definitions of computational thinking, finding they share a lot of common elements, of very different nature. We classified them in mental processes, methods, practices, and transversal skills. Many of these elements seem to be shared with other disciplines and resonate with the current narrative on the importance of 21st-century skills. Our classification helps on shedding light on the misconceptions related to each of the four categories, showing that, not to dilute the concept, elements of computational thinking should be intended inside the discipline of Informatics, being its "disciplinary way of thinking".

Keywords: computational thinking, informatics, misconceptions, definition, disciplinary way of thinking, informatical thinking.

1. Introduction[1]

The expression *computational thinking* (CT, from now on) seems to have been firstly used in print by Seymour Papert (1980) and then was brought to the attention of the informatics community by Jeannette Wing (2006).

From 2006, a considerable body of literature has been produced to search for a better definition of this concept, to provide tools and frameworks, to introduce and assess CT in K-12 education.

Even if there is no agreement between authors, a lot of proposed definitions stress the fact that CT is not only about technical methods and practices, but also about mental processes and transversal competences[2] like *creativity, collaboration, tolerance for ambiguity, resilience,* and more. However, educational and psychological research warns about optimistic claims on the transfer of competences from a discipline to other far domains and to general skills.

---

[1] This paper is based on material from author's PhD thesis (Lodi, 2020).

[2] Often referred also as *transversal skills*, *soft skills* or *key competences*, in the context of EU documents, in particular in the "*Personal, social and learning competence,*" see for example
`http://data.consilium.europa.eu/doc/document/ST-5464-2018-ADD-2/EN/pdf`

In this paper, we review some of the most important definitions emerged in the last years and propose a classification of the common elements that can be useful to better frame the misinterpretations of the concepts.

We will argue that CT must maintain its bond with informatics, representing its "disciplinary way of thinking".

2. Definitions of Computational Thinking[3]

The expression "computational thinking" was brought back to the informatics community by Wing (2006), gaining massive attention[4]. In that seminal article, Wing did not give a definition, but related the concept to informatics, stating "*Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science*", arguing that "*thinking like computer scientists*" would be a benefit for everyone, not just for professionals or scientists.

In these years, many definitions have been proposed. In Corradini *et al.* (2017b) authors started from five of the most important definitions to find out constituting elements of CT. Juškevičienė & Dagienė (2018) schematized many of the definitions proposed from Papert's views up to 2017. We review all of them in the following table. In the third column, we provide pointers to the classification that we will propose in Section 3.

Paper	Description	Elements (Sec. 3)
Wing (2006)	In her seminal article, Wing informally defines CT as *"thinking like computer scientists"*.	1)
Wing (2008, 2011)	Wing defines more formally CT as "*the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent*".	1) d) 2) a)
	This definition is attributed to Jan Cuny, Larry Snyder, and Jeannette M. Wing, in an unpublished work from 2010: "Demystifying Computational Thinking for Non-Computer Scientists," referenced by Wing (2011) herself. Moreover, Wing says it was originated by a discussion with Alfred Aho.	1) a) b) 2) c) e) f)
	Wing also identifies characteristic elements of CT. In particular, she states the most important elements are *abstraction* (the "mental" tool of computing) and *automation* (by using a computer, the "metal" tool of computer scientists): "*computing is the automation of our abstractions*" (Wing, 2008).	
	Wing (2011) recognises significant overlapping or inclusions between CT and other types of thinking: logical thinking, algorithmic thinking, parallel thinking, compositional reasoning, pattern matching, procedural thinking, and recursive thinking.	

Continued on next page

---

[3] This section and the following one (Sec. 3) is an expansion of part of the work presented in Corradini *et al.* (2017b).

[4] Currently (July 2020), the paper has more than 6300 citations, according to Google Scholar.

Table – continued from previous page

Paper	Description	Elements (Sec. 3)
Aho (2011)	Alfred Aho provides a definition very similar to the Cuny-Snyder-Wing one, more focused on "algorithmic thinking": "*We consider computational thinking to be the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms.*" It is worth noticing that Aho stresses, in particular, the role played in this definition by the information processing agent, and that computational thinking should be based on clearly defined models of computation.	1) a)
ISTE&CSTA (2011a,b)	ISTE & CSTA proposed an operational definition, targeting specifically K-12 educators. They define CT as a problem-solving process that includes (but is not limited to) the following characteristics: • *Formulating problems in a way that enables us to use a computer and other tools to help solve them* • *Logically organizing and analyzing data* • *Representing data through abstractions such as models and simulations* • *Automating solutions through algorithmic thinking (a series of ordered steps)* • *Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources* • *Generalizing and transferring this problem-solving process to a wide variety of problems* Moreover, they state that CT is "*supported and enhanced by a number of dispositions or attitudes*" that include: • *Confidence in dealing with complexity* • *Persistence in working with difficult problems* • *Tolerance for ambiguity* • *The ability to deal with open ended problems* • *The ability to communicate and work with others to achieve a common goal or solution* Finally they propose a CT vocabulary (ISTE & CSTA, 2011a), listing a set of CT terms with a brief definition or explanation: • *Data Collection* • *Data Analysis* • *Data Representation* • *Problem Decomposition* • *Abstraction* • *Algorithms and Procedures* • *Automation* • *Simulation* • *Parallelization*	1) a) c) d) f) 2) a) b) c) d) e) 4) b) d) e)
Google (n.d.)	Google assumes the same ISTE/CSTA definition but – instead of a vocabulary – lists and (re)defines a series of CT concepts, pointing out that they are *mental processes* or *tangible outcomes*: • *Abstraction* • *Algorithm Design* • *Automation* • *Data Analysis* • *Data Collection* • *Data Representation* • *Decomposition* • *Parallelization* • *Pattern Generalization* • *Pattern Recognition* • *Simulation*	1) a) c) d) e) f) 2) a) b) c) d)

Table – continued from previous page

Paper	Description	Elements (Sec. 3)
Brennan & Resnick (2012)	Brennan and Resnick present a *computational thinking framework* to describe learning and development that take place when designing and programming interactive media with Scratch platform. They state CT involves three dimensions:	1) a) c) d)
	• *Computational concepts* designers employ as they program:	2) b) c) f)
	◦ *Sequences*	
	◦ *Loops*	
	◦ *Parallelism*	3) a) b) c)
	◦ *Events*	
	◦ *Conditionals*	
	◦ *Operators*	4) a) b) c)
	◦ *Data*	
	• *Computational practices* designers develop as they program:	
	◦ *Being incremental and iterative,*	
	◦ *Testing and debugging,*	
	◦ *Reusing and remixing*	
	◦ *Abstracting and modularizing*	
	• *Computational perspectives* designers form about the world around them and about themselves:	
	◦ *Expressing*	
	◦ *Connecting*	
	◦ *Questioning*	
Csizmadia *et al.* (2015)	Computing at school assume a Wing-like definition: CT is "*learning to think in ways which allow us, as humans, to solve problems more effectively and, when appropriate, use computers to help us do so*" and then state it involves	1) a) b) c) d) e) f)
	• Six concepts	
	◦ *Logic*	2) e)
	◦ *Algorithms*	
	◦ *Decomposition*	3) a) b)
	◦ *Patterns*	
	◦ *Abstraction*	4) a) b)
	◦ *Evaluation*	e)
	• Five approaches	
	◦ *Tinkering*	
	◦ *Creating*	
	◦ *Debugging*	
	◦ *Persevering*	
	◦ *Collaborating*	
Grover and Pea (2013)	After a literature review, assumed the Aho-Cuny-Snyder-Wing definition and agreed that the following elements are accepted:	1) a) b) c) d) f)
	• *Abstractions and pattern generalizations (including models and simulations)*	
	• *Systematic processing of information*	2) b) c) d) e) f)
	• *Symbol systems and representations*	
	• *Algorithmic notions of flow of control*	
	• *Structured problem decomposition (modularizing)*	3) b)
	• *Iterative, recursive, and parallel thinking*	
	• *Conditional logic*	
	• *Efficiency and performance constraints*	
	• *Debugging and systematic error detection*	

Table – continued from previous page

Paper	Description	Elements (Sec. 3)
Selby and Woollard (2013)	In a widely referenced Technical Report, Selby and Woollard examined a number of CT definitions, and argued that the most relevant and useful elements are: • *Thought process* • *Abstraction* • *Decomposition* • *Algorithmic thinking* • *Evaluation* • *Generalization*	1) a) c) d) f) 2) e)

Weintrop *et al.* (2016)	They propose a definition of CT for mathematics and science. From a literature review, they start with an initial set of activities:
	• *Ability to deal with open-ended problems*
	• *Persistence in working through challenging problems*
	• *Confidence in dealing with complexity*
	• *Representing ideas in computationally meaningful ways*
	• *Breaking down large problems into smaller problems*
	• *Creating abstractions for aspects of problem at hand*
	• *Reframing problem into a recognizable problem*
	• *Assessing strengths/weaknesses of a representation of data/representational system*
	• *Generating algorithmic solutions*
	• *Recognizing and addressing ambiguity in algorithms*
	After that, by analyzing CT activities for math and science, propose the "Computational thinking in mathematics and science taxonomy".
	• *Data practices:*
	  ◦ *Collecting Data*
	  ◦ *Creating Data*
	  ◦ *Manipulating Data*
	  ◦ *Analyzing Data*
	  ◦ *Visualiging Data*
	• *Modeling and simulation practices:*
	  ◦ *Using Computational Models to Understand a Concept*
	  ◦ *Using Computational Models to Find and Test solutions*
	  ◦ *Assessing Computational Models*
	  ◦ *Designing Computational Models*
	  ◦ *Constructing Computational Models*
	• *Computational problem solving practices:*
	  ◦ *Preparing Problems for Computational Solutions*
	  ◦ *Programming*
	  ◦ *Choosing Effective Computational Tools*
	  ◦ *Assessing Different Approaches/Solutions to a Problem*
	  ◦ *Developing Modular Computational Solutions*
	  ◦ *Creating Computational Abstractions*
	  ◦ *Troubleshooting and Debugging*
	• *Systems thinking practices:*
	  ◦ *Investigating a Complex System as a Whole*
	  ◦ *Understanding the Relationships within a System*
	  ◦ *Thinking in levels*
	  ◦ *Communicating Informations about a System*
	  ◦ *Defining Systems and Managing Complexity*
		1) a) d) e) f)
		
		2) b) d) e) f)
		
		3) b)
		
		4) a) d) e)

Table – continued from previous page

Paper	Description	Elements (Sec. 3)
Kalelioğlu *et al.* (2016)	They view CT as a "*complex higher-order thinking, skills may require to use the power of human cognitive ability and embrace the support of machines to think and solve problems.*" They propose a "Framework for Computational Thinking as a Problem Solving Process" in five steps.	1) c) d) e) f)
	• *Identify the problem*	2) a) b) c) d) e)
	◦ *Abstraction*	
	◦ *Decomposition*	3) b)
	• *Gathering, representing and analysing data*	
	◦ *Data collection*	4) a)
	◦ *Data analysis*	
	◦ *Pattern recognition*	
	◦ *Conceptualising*	
	◦ *Data representation*	
	• *Generate, select and plan solutions*	
	◦ *Mathematical reasoning*	
	◦ *Building algorithms and procedures*	
	◦ *Parallelisation*	
	• *Implement solutions*	
	◦ *Automation*	
	◦ *Modelling and simulations*	
	• *Assessing solutions and continue for improvement*	
	◦ *Testing*	
	◦ *Debugging*	
	◦ *Generalisation*	
Krauss and Prottsman (2016)	Krauss and Prottsman define[5] CT as *using thinking patterns and processes to pose and solve problems or prepare programs for computation*. Lessons plans are given for the following categories:	1) a) c) d) e)
	• *Decomposition (data analysis)*	2) a) b) c) d)
	• *Pattern matching (data visualization)*	
	• *Abstraction (data modelling, pattern generalization)*	
	• *Automation (algorithm design, parallelization, simulation)*	4) a)
Shute *et al.* (2017)	After an extensive literature review, they provide a very general definition of CT: "*the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts.*". They then recognize the following categories and subcategories, giving however explanations that are quite general and far from Informatics.	1) a) c) d) e) f)
		2) a) b) c) d) e)
	• *Decomposition*	3) a) b)
	• *Abstraction (data collection and analysis, pattern recognition, modelling)*	
	• *Algorithms (algorithm design, parallelism, efficiency, automation)*	4) a)
	• *Debugging*	
	• *Iteration*	
	• *Generalization*	

---

[5] As cited in Juškevičienė and Dagienė (2018, p. 270).

Table – continued from previous page

Paper	Description	Elements (Sec. 3)
College Board (2017)	Proposes a CT framework for the *AP CS Principles* course, made of six practices.	1) a) d)
	• *Connecting Computing*	
	◦ *Identify impacts of computing.*	2) b)
	◦ *Describe connections between people and computing.*	d) e)
	◦ *Explain connections between computing concepts.*	
	• *Creating Computational Artifacts*	3) b)
	◦ *Create a computational artifact with a practical, personal, or societal intent.*	
	◦ *Select appropriate techniques to develop a computational artifact.*	4) a) b)
	◦ *Use appropriate algorithmic and information management principles.*	c)
	• *Abstracting*	
	◦ *Explain how data, information, or knowledge is represented for computational use.*	
	◦ *Explain how abstractions are used in computation or modeling.*	
	◦ *Identify abstractions.*	
	◦ *Describe modeling in a computational context.*	
	• *Analyzing Problems and Artifacts*	
	◦ *Evaluate a proposed solution to a problem.*	
	◦ *Locate and correct errors.*	
	◦ *Explain how an artifact functions.*	
	◦ *Justify appropriateness and correctness of a solution, model, or artifact.*	
	• *Communicating*	
	◦ *Explain the meaning of a result in context.*	
	◦ *Describe computation with accurate and precise language, notations, or visualizations.*	
	◦ *Summarize the purpose of a computational artifact.*	
	• *Collaborating*	
	◦ *Collaborate with another student in solving a computational problem.*	
	◦ *Collaborate with another student in producing an artifact.*	
	◦ *Share the workload by providing individual contributions to an overall collaborative effort.*	
	◦ *Foster a constructive, collaborative climate by resolving conflicts and facilitating the contributions of a partner or team member.*	
	◦ *Exchange knowledge and feedback with a partner or team member.*	
	◦ *Review and revise their work as needed to create a high-quality artifact.*	
Denning and Tedre (2019)	Denning and Tedre, in their book about CT, proposed the following definition: *Computational thinking is the mental skills and practices for*	4) a) c)
	• *Designing computations that get computers to do jobs for us, and*	2) a) d)
	• *Explaining and interpreting the world as a complex of information processes*	
	Moreover, they distinguish between "CT for beginners" (the one that is spreading in K-12 education, teaching basic computational problem solving) with "CT for professionals" (the one used by cutting-edge engineers and scientists in all fields as a powerful professional tool).	
	They recognize CT has six important dimensions, "windows" looking at CT:	
	• *Methods*	
	• *Machines*	
	• *Computing Education*	
	• *Software Engineering*	
	• *Design*	
	• *Computational Science*	

Table – continued from previous page

Paper	Description	Elements (Sec. 3)
Juškevičienė & Dagienė (2018)	After reviewing many definitions, Juškevičienė & Dagienė found eight CT components groups.	1) a) b) c) d) e) f)
	• *Data analysis & representation*	
	◦ *Data collection*	
	◦ *Data analysis*	2) a) b)
	◦ *Data representation*	d) e)
	◦ *Generalisation*	
	◦ *Patterns finding*	3) b)
	◦ *Drawing conclusions*	
	• *Computing Artefacts*	4) a) b)
	◦ *Artefact development*	c)
	◦ *Artefact designing*	
	• *Decomposition*	
	◦ *Breaking into parts*	
	• *Abstraction*	
	◦ *Details suppression*	
	◦ *Modelling*	
	◦ *Information filtering*	
	• *Algorithms*	
	◦ *Sequence of steps*	
	◦ *Procedures*	
	◦ *Set of instructions*	
	◦ *Automation*	
	• *Communication & collaboration*	
	◦ *Communication*	
	◦ *Collaboration*	
	◦ *Computational analysis*	
	• *Computing & Society*	
	◦ *Computing influence*	
	◦ *Computing implication*	
	◦ *Computing concepts*	
	• *Evaluation*	
	◦ *Evaluation*	
	◦ *Correction*	

3. Comparison

We compared the CT elements found in the analysed definitions.

Those who give a precise definition agree on the fact that **CT is a way of thinking** (*thought process*) for **problem solving**. They all somehow specify that it is a ***computational*** (rather than general) *problem solving*: the formulation and the solution of the problem must be expressed in a way that allows an "external" processing agent (a human or a machine) to carry it out.

Apart from the general statement, all definitions list some constitutive elements of CT. These elements are of very different kinds (from thinking habits to specific program-

ming concepts), and many authors group them in categories, but there is no universal agreement on the classification.

We classified all the elements into four categories. For each category, we list the elements, trying to summarise all aspects stated in the analysed definitions. Instead of an "intersection approach", keeping only the elements that had a wider consensus (like what was done by Selby & Woollard (2013)), we used a "union approach", trying to build a comprehensive list of all the characteristics proposed by different authors.

1) **Mental/thought processes**: mental strategies useful to solve problems.

   a) *Algorithmic thinking*: use algorithmic/procedural thinking (ISTE & CSTA, 2011b; Wing, 2008, 2011) to design a sequence of ordered step (instructions) to solve a problem, achieve a goal or perform a task (Brennan & Resnick, 2012; Csizmadia *et al.*, 2015; Google, n.d.; ISTE & CSTA, 2011a). Also recognised by (College Board, 2017; Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Krauss & Prottsman, 2016; Selby & Woollard, 2013; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

   b) *Logical thinking*: use logical thinking (Wing, 2011) and reasoning to make sense of things, establish and check facts (Csizmadia *et al.*, 2015). Also recognised by (Grover & Pea, 2013; Juškevičienė & Dagienė, 2018).

   c) *Problem decomposition and modularisation*: split a complex problem into simpler subproblems to solve it more easily (Csizmadia *et al.*, 2015; Google, n.d.; ISTE & CSTA, 2011a); modularise (Brennan & Resnick, 2012); use compositional reasoning (Wing, 2008). Also recognised by (Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Selby & Woollard, 2013; Shute *et al.*, 2017).

   d) *Abstraction*: get rid of useless details to focus on relevant information or ideas (Brennan & Resnick, 2012; Csizmadia *et al.*, 2015; Google, n.d.; ISTE & CSTA, 2011a; Wing, 2011). Also recognised by (College Board, 2017; Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Selby & Woollard, 2013; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

   e) *Pattern recognition*: discover and use regularities in data and problems (Csizmadia *et al.*, 2015; Google, n.d.; Wing, 2011). Also recognised by (Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

   f) *Generalisation*: use discovered similarities to make predictions or to solve more general problems (Csizmadia *et al.*, 2015; Google, n.d.; ISTE & CSTA, 2011b). Also recognised by (Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Selby & Woollard, 2013; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

2) **Methods**: operational approaches widely used by computer scientists.

   a) *Automation*: automate the solutions (ISTE & CSTA, 2011b; Wing, 2008); use a computer or a machine to do repetitive tasks (Google, n.d.; ISTE & CSTA, 2011a). Also recognised by (Denning & Tedre, 2019; Juškevičienė

& Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Shute *et al.*, 2017).

b) *Data collection, analysis and representation*: gather information/data, make sense of them by finding patterns, represent them properly (Google, n.d.; ISTE & CSTA, 2011a); store, retrieve and update values (Brennan & Resnick, 2012). Also recognised by (College Board, 2017; Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

c) *Parallelisation*: carry out tasks simultaneously to reach a common goal (Brennan & Resnick, 2012; Google, n.d.; ISTE & CSTA, 2011a), use parallel thinking (Wing, 2011). Also recognised by (Grover & Pea, 2013; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Shute *et al.*, 2017).

d) *Modelling and simulation*: represent data and (real world) processes through models (Google, n.d.; ISTE & CSTA, 2011b), run experiments on models (ISTE & CSTA, 2011a). Also recognised by (College Board, 2017; Denning & Tedre, 2019; Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

e) *Analysis and evaluation*: implement and analyse solutions (ISTE & CSTA, 2011a) to judge them (Csizmadia *et al.*, 2015), in particular for what concerns effectiveness, and efficiency in terms of time and resources. Also recognised by (College Board, 2017; Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Selby & Woollard, 2013; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

f) *Programming*: use some common concepts in programming (e.g. loops, events, conditionals, mathematical and logical operators (Brennan & Resnick, 2012), recursion (Wing, 2011)). Also recognised by (Grover & Pea, 2013; Weintrop *et al.*, 2016).

3) **Practices**: typical practices used in the implementation of computing machinery based solutions.

a) *Experimenting, iterating, tinkering*: in iterative and incremental software development, one develops a project with repeated iterations of a design-build-test cycle, incrementally building the final result (Brennan & Resnick, 2012); tinkering means trying things out using a trial and error process, learning by playing, exploring, and experimenting (Csizmadia *et al.*, 2015). Also recognised by (Shute *et al.*, 2017).

b) *Test and debug*: verify that solutions work by trying them out (Brennan & Resnick, 2012); find and solve problems (bugs) in a solution/ program (Csizmadia *et al.*, 2015). Also recognised by (College Board, 2017; Grover & Pea, 2013; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

c) *Reuse and remix*: build your solution on existing code, projects, ideas (Brennan & Resnick, 2012).

4) **Transversal skills**: general ways of seeing and operating in the world fostered by thinking like computer scientists; useful life skills that *can enhance* thinking like a computer scientist.

   a) *Design and create*: design and build things (Csizmadia *et al.*, 2015) and computational artifacts, use computation to be creative and express yourself (Brennan & Resnick, 2012). Also recognised by (College Board, 2017; Denning & Tedre, 2019; Juškevičienė & Dagienė, 2018; Kalelioğlu *et al.*, 2016; Krauss & Prottsman, 2016; Shute *et al.*, 2017; Weintrop *et al.*, 2016).

   b) *Communicate and collaborate*: connect with others and work together to create something with a common goal and to ensure a better solution (Brennan & Resnick, 2012; Csizmadia *et al.*, 2015; ISTE & CSTA, 2011b). Also recognised by (College Board, 2017; Juškevičienė & Dagienė, 2018).

   c) *Reflect, learn, meta-reflect, understand the world computationally*: use computation to reflect and understand computational aspects of the world (Brennan & Resnick, 2012), identify impacts of computing on society (College Board, 2017). Also recognised by (Denning & Tedre, 2019; Juškevičienė & Dagienė, 2018).

   d) *Be tolerant for ambiguity*: deal with non-well specified and open-ended, real-world problems(ISTE & CSTA, 2011b). Also recognised by (Weintrop *et al.*, 2016).

   e) *Be persistent when dealing with complex problems*: be confident in working with difficult or complex problems (ISTE & CSTA, 2011a), persevering, being determined, resilient and tenacious (Csizmadia *et al.*, 2015). Also recognized by (Weintrop *et al.*, 2016).

4. CT as Informatics "Disciplinary Way of Thinking"

Note that many of the cited elements in the previous classification are broad and general. This led to some critiques (for an overview, see Martins-Pacheco *et al.* (2020)): some of these concepts are not exclusively associated with Informatics, but taught in other disciplines (e.g., Math and Sciences) or are general skills that children have been learning for a long time before the birth of Informatics. Cansu & Cansu (2019), summarising critics from Hemmendinger (2010), stated for example that:

- *Reformulating hard problems is typical of all domains of problem solving.*
- *Philosophers have been thinking about thinking – recursively – for a long time.*
- *Mathematics surely uses abstraction, and so do all disciplines that build models.*
- *Separation of concerns and using heuristics also characterizes problem-solving in general.*

By contrast, other authors argue that computing features extend and differentiate these elements from other domains (Grover & Pea, 2013), and provide some characteristic problem-solving methods (e.g., the possibility to effectively execute a solution, a model, an abstraction by running an implementation of its algorithm (Martini, 2012)).

We agree, arguing that Informatics is what needs to be taught in schools, and CT is, at most, the *conceptual sediment* of that teaching, what remains even when the technical aspects have been forgotten (Lodi *et al.*, 2017).

In other disciplines, like Math, it is recurrent to talk about mathematical thinking (Sternberg & Ben-Zeev, 1996), or mathematical reasoning (English, 1997), or mathematical problem solving (Schoenfeld, 1985).

Like what mathematical thinking is for Math, CT is Informatics "disciplinary way of thinking" (Pace & Middendorf, 2004) (and this explains the provocative title "Informatical thinking").

Chick *et al.* (2009), talking about *signature pedagogies* (pedagogies to "engage students in the ways of knowing, the habits of mind, and the values shared by experts in [a] field" (Gurung *et al.*, 2009, p. xvii)), affirms that "effective teaching results from core values and principles of our courses and of our disciplines, rather than from generic views of learning. […] higher-level thinking is inhibited by such generic conceptions and lays the groundwork for questions about the central values, habits, and ways of thinking within their disciplines" (Chick *et al.*, 2009, p. 4).

According to Li *et al.* (2019, p. 8)

> […] domain-specific thinking and domain-general thinking are not dichotomous, as thinking itself is a complex process involving many different components. Domain-general thinking is often derived from human's thinking performance across different knowledge-lean (e.g., solving a puzzle) or -rich task domains (e.g., solving algebraic equations). Domain-specific thinking is often characterized in terms of its disciplinary content but also involves more general cognitive components. In other words, domain-specific thinking should contain both domain-specific and -general aspects of cognitive activities. For example, a mathematician's thinking is scarcely only mathematics (the knowledge component). It can share possible common elements with a biologist's thinking (e.g., certain aspects of metacognition and meta-representation). The same reasoning applies to students' thinking in specific disciplines. […] some aspects of mathematical problem solving are largely discipline specific (e.g., the knowledge base), some heavily discipline-oriented (e.g., strategies and beliefs), some much like discipline domain-general (e.g., metacognition).

While we agree disciplinary thinking contains both specific and general aspects, we keep spotting the tendency of educators and policymakers to focus, for what concerns CT, only on the more general ones.

Voogt *et al.* (2015) recognise, in some of the abovementioned definitions of CT, a tension between "*the 'core' qualities of CT versus certain more 'peripheral' qualities*". The latter highly overlap with what we called "transversal skills," and we agree with

Voogt that this "*runs the risk of diluting the idea of CT, blurring and making it indistinct from other 21st century skills*".

As CT movement has grown in educational contexts, many unverified claims about the effects of learning CT/Informatics has emerged (e.g., that it will *automatically* transfer to thinking logically, better problem solving in every aspect of life, developing perseverance, getting better results in math and science, and so on (Lewis, 2017)). Most of these claims are not supported by research, and "*appear in blog posts, opinion pieces, and other 'grey literature'*" (Duncan, 2019, p. 32). As Denning & Tedre (2019, p.xiii) put it:

> [CT] is sometimes portrayed as a universal approach to problem solving. Take a few programming courses, the story in the popular media goes, and you will be able to solve problems in any field. Would that this were true! Your ability to solve a problem for someone depends on your understanding of their context in which the problem exists. For instance, you cannot build simulations of aircraft in flight without understanding fluid dynamics. You cannot program searches through genome databases without understanding the biology of the genome and the methods of collecting the data. Computational thinking is powerful, but not universal.

As we will see, non-specialist teachers that most probably never studied Informatics in their schooling or training may tend to stick to some "general versions" of the listed characteristics (and especially on the peripheral qualities), not necessarily related to Informatics.

We believe, by contrast, CT must be understood *inside* Informatics: while many characteristics are (more or less apparently) shared with other disciplines, we need to focus on their specific "informatical" instantiation.

We believe that the classification we proposed in the previous section is a good tool to frame the misconceptions about CT and Informatics in K-12 education (Denning, 2017; Denning *et al.*, 2017). In the next four subsections, we will discuss in this light the four main categories we recognised.

4.1. *Mental Processes*

Mental processes are, on the surface, shared with other disciplines, but should be understood and experienced as Informatics specific. "CT draws on a rich legacy of related frameworks as it extends previous thinking skills" (Lee *et al.*, 2011, p. 32).

First of all, analysed definitions are clear in stressing on the *computational* (rather than *general*) nature of *problem solving*. In facts, many authors (recall for example Aho's position) agree that what differentiates algorithmic thinking (which has been used for centuries, firstly by mathematicians) and computational thinking is the *automation* of the algorithm (Stephens & Kadijevich, 2020).

Next, as diSessa points out, *abstraction*, one of the core CT concepts according to Wing and many others, has different nuances in different disciplines:

> Mathematical abstraction (let's call it, inferential abstraction) occurs in order to build conceptual worlds where a small set of attributes fully define entities, resulting in a substantial inferential fabric – a family of basic ideas and secure inferences (proofs) from them to other ideas (theorems). Abstraction in physics (empirical abstraction) is taking a look at the world and finding in it new things that cut away certain details, but build on others that might initially be completely ignored, in order to create core models that apply across a very wide range of circumstances. Mathematicians do not, in general, need or use the skill of "peeling away" from the world as it exists, nor digging through the difficulties of finding out how the world is in the first place, nor do they have the constraint of confirming empirically that the world admits in a certain abstraction, usually within prescribed limits. Abstraction in computer science (practical abstraction) resides substantially in peeling away the irrelevant particulars of an implementation so that one only need think about the features of a piece of it that are essential for a given use – its "contract" with the rest of the program. (diSessa, 2018, p. 21)

Moreover, it is hugely debated if general skills (like *general problem solving, critical thinking, creative thinking, decision making, and so on*) exist, are transferrable or even teachable (for a comprehensive review, see (Lodi, 2020, ch. 4)). For example, Gick & Holyoak (1980) found that *problem decomposition*, one of the most highlighted aspects of CT (Guzdial, 2019), is not easily transferrable. They described to students a situation where an army had to be divided into small groups to successfully attack a fortress; immediately after they asked the students how to attack a tumour with a laser without damaging healthy tissues. The vast majority of the students were not able to use the same approach (divide the laser in multiple weaker beams). They only managed to do so when explicitly prompted to think at the army example.

That is why these skills should be taught in an Informatics-specific context.

4.2. *Informatics Methods*

Many methods are, again, shared with other disciplines, but we believe they must be experienced in the context of automatic elaboration of information.

Emblematic examples are *unplugged activities*, teaching activities not using a computer or tablet or smartphone to teach informatics concepts and methods. After experiencing the activities without computers, it is essential to relate what students have done to the specific informatics context, to understand what happens on physical devices. Bell & Vahrenhold (2018) suggest that Unplugged activities offer best results when used

in combination with "plugged approaches" (i.e. programming tasks). Using unplugged activities before the plugged one seems to foster even *more* effective results than the programming activities alone.

Two early studies discovered that, without this explicit connection, "*the program* [based on CS Unplugged] *had no statistically significant impact on student attitudes toward computer science or perceived content understanding*" (Feaster *et al.*, 2011) and that "*the students' attitudes and intentions regarding CS did not change in the desired direction*" (Taub *et al.*, 2012).

4.3. *Informatics Practices*

Listed practices are, of course, shared with other disciplines and activities. As we will also discuss for transversal competences, we should not justify the introduction of Informatics in K-12 education mainly for teaching this kind of general approaches (which have been used and taught for centuries before the advent of Informatics).

However, we agree that computers are powerful tools to "concretely experiment with". This is one aspect of the constructionist learning theory from Seymour Papert (see Lodi (2020, chapters 3, 6)).

Already in 1970, Papert and colleagues, while designing and experimenting with the LOGO programming language, noted (Feurzeig *et al.*, 1970) that the peculiarities of computer programming make it a privileged tool for learning problem solving with an experimental approach. In fact, children have to impose on themselves rigour and precision in instructing the computer – being explicit and precise is not imposed (incomprehensibly) by enforcement of the teacher, but naturally emerges from the need of being understood by an automated executor with a limited instruction set, which is unable to perform any "human" inference. Briefly: the computer creates an intrinsic motivation to learn by trial, error and debug.

4.4. *Transversal Competences*

Transversal competences like perseverance and tolerance for ambiguity are useful for learning a difficult topic like Informatics, but including it in the definition may cause people to think CT is mainly about these competences.

For example, Corradini *et al.* (2017a) found that teachers saw the value of Programma il Futuro project (the Italian version of Code.org) more in fostering transversal competences or domain-general skills (like promotion of awareness and comprehension of problem solving, logical thinking, creativity, attention, planning ability, motivation for learning, students interest, cooperation) than in teaching Informatics core concepts.

The same sample (Corradini *et al.*, 2017b) struggled to give a sound and complete definition of CT, focusing on some crucial aspects like problem solving, mental pro-

cesses, logic, but often forgetting to refer in any way to an *information-processing agent*, giving a very partial view of Informatics. Moreover, many of them mentioned transversal competences, which hints the view of Informatics as an instrumental discipline, not valuable in itself. This is possibly deriving from attempts to convince teachers of the importance of CT by focusing mainly on its value for other disciplines and as a general learning tool.

Moreover, non-specialists may get the wrong direction of the implication: while researchers agree that competences like perseverance, dealing with complexity, and collaboration are essential to succeed in Informatics, which is a challenging subject (Murphy & Thomas, 2008), the opposite implication (CT fostered by these skills) is far from being proved.

For example, Lewis (2017, p. 18) states that "*programming has been speculated to be uniquely qualified to help normalize failure and thus encourage productive learning strategies*". However, research in education tells us that transfer is difficult and unlikely to happen, especially between knowledge domains far from one another, and especially when treating domain-general skills (Guzdial, 2015; Lewis, 2017).

At the moment, there is no proof that transversal skills like perseverance are automatically fostered by learning CT (for examples, we found no difference in students' mindset, with respect of studying or not studying Informatics at school (Lodi, 2019)).

5. Conclusions

The expression "computational thinking" has become a buzzword related to the introduction of CS in K-12 education. Although it had already been used in the 80s by Papert, it started to be massively used in Informatics education after being re-proposed by Wing.

Many authors tried to define CT: despite being quite different, the most famous definitions share many characteristics. All agree CT is a form of thinking for solving problems by expressing the solution in a way that can be automatically carried out by an (external) processing agent. We identified four categories of CT constitutive elements proposed by authors: mental processes, methods, practices, and transversal skills. We argued that this classification could be useful to frame misconceptions about CT.

Mental processes (e.g., problem solving, problem decomposition, abstraction, logical thinking) and transversal competences (e.g., tolerance for ambiguity, perseverance) resonate with the current narrative on the importance of the 21st-century skills, and are probably even one of the reasons of the widespread of CT in education. This is confirmed by large scale qualitative studies (Corradini *et al.*, 2017a), showing that generalist teachers mainly find value in introducing CT in schools for promoting general skills rather than Informatics core concepts.

However, educational research warns about the transferability of this kind of general skills between disciplines, and some even doubt their teachability.

Moreover, putting too much focus on this aspects risks to dilute the fundamental concepts that distinguish CS from other disciplines (e.g., the presence of a precise external executor that solves problems following provided algorithms, the possibility to describe and execute abstractions through specific languages, the possibility to simulate worlds, and so on). The definitions of CT contain many elements directly linked to CS methods (e.g., automation, data analysis, evaluation) and programming practices (e.g., iterating, debugging). However, educators may fail to include references to these CS specific aspects in their definition of CT.

Since even more specific concepts appear to be shared with other disciplines, the message that teaching separately some of these concepts (often in an informatics-unrelated way) – or simply recognising them inside other disciplines (like math for problem solving, geography for giving precise directions, and so on) – will automatically foster CT is spreading between educators.

We therefore argued that all the characteristics, especially the "core" ones, should be read, understood, and taught *inside* the discipline of Informatics. A lot of "thinking" are worth being taught, CT "*is often a welcome addition to other fields, but not a replacement for their ways of thinking and not a meta-skill for all fields*" (Denning & Tedre, 2019, p. 213).

CT should represent the "disciplinary way of thinking" of Informatics: *Informatical thinking*.

Acknowledgements

References

Aho, A. V. (2011). Ubiquity Symposium: Computation and Computational Thinking. Ubiquity, 2011(January). https://doi.org/10.1145/1922681.1922682

Bell, T., & Vahrenhold, J. (2018). CS Unplugged – How Is It Used, and Does It Work? In H.-J. Böckenhauer, D. Komm, & W. Unger (Eds.), Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday (pp. 497–521). Springer International Publishing. https://doi.org/10.1007/978-3-319-98355-4_29

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking (Using artifact-based interviews to study the development of computational thinking in interactive media design). Proceedings of the 2012 Annual Meeting of the American Educational Research Association. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Cansu, S. K., & Cansu, F. K. (2019). An Overview of Computational Thinking. International Journal of Computer Science Education in Schools, 3(1). https://eric.ed.gov/?id=EJ1214682

Chick, N. L., Haynie, A., Gurung, R. A., & Regan, A. (2009). From generic to signature pedagogies: Teaching disciplinary understandings. In R. A. R. Gurung, N. L. Chick, & A. Haynie (Eds.), Exploring signature pedagogies: Approaches to disciplinary habits of mind. (pp. 1–16). Stylus publishing. https://books.google.co.il/books?id=0SWec-nwL4EC

College Board. (2017). AP Computer Science Principles. College Board. `https://apcentral.college-board.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf`

Corradini, I., Lodi, M., & Nardelli, E. (2017a). Computational Thinking in Italian Schools: Quantitative Data and Teachers' Sentiment Analysis after Two Years of "Programma Il Futuro". Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, 224–229. `https://doi.org/10.1145/3059009.3059040`

Corradini, I., Lodi, M., & Nardelli, E. (2017b). Conceptions and Misconceptions about Computational Thinking among Italian Primary School Teachers. Proceedings of the 2017 ACM Conference on International Computing Education Research, 136–144. `https://doi.org/10.1145/3105726.3106194`

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking – A guide for teachers. Computing at School. `https://eprints.soton.ac.uk/424545/`

Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. Commun. ACM, 60(6), 33–39. `https://doi.org/10.1145/2998438`

Denning, P. J., & Tedre, M. (2019). Computational Thinking. MIT Press.

Denning, P. J., Tedre, M., & Yongpradit, P. (2017). Misconceptions about Computer Science. Commun. ACM, 60(3), 31–33. `https://doi.org/10.1145/3041047`

diSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. Mathematical Thinking and Learning, 20(1), 3–31. `https://doi.org/10.1080/10986065.2018.1403544`

Duncan, C. (2019). Computer science and computational thinking in primary schools. [PhD Thesis, University of Canterbury]. `http://hdl.handle.net/10092/17160`

English, L. D. (1997). Mathematical Reasoning: Analogies, Metaphors, and Images. L. Erlbaum Associates.

Feaster, Y., Segars, L., Wahba, S. K., & Hallstrom, J. O. (2011). Teaching CS Unplugged in the High School (with Limited Success). Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, 248–252. `https://doi.org/10.1145/1999747.1999817`

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1970). Programming-Languages as a Conceptual Framework for Teaching Mathematics. SIGCUE Outlook, 4(2), 13–17. `https://doi.org/10.1145/965754.965757`

Gick, M. L., & Holyoak, K. J. (1980). Analogical problem solving. Cognitive Psychology, 12(3), 306–355. `https://doi.org/10.1016/0010-0285(80)90013-4`

Google. (n.d.). Exploring Computational Thinking. `http://g.co/exploringct` – The page has now been removed, but can be found in the "CT overview" tab here: `https://web.archive.org/web/20181001115843/https://edu.google.com/resources/programs/ exploring-computational-thinking/`

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43. `https://doi.org/10.3102/0013189X12463051`

Gurung, R. A. R., Chick, N. L., & Haynie, A. (Eds.). (2009). Exploring Signature Pedagogies: Approaches to Teaching Disciplinary Habits of Mind. Stylus Publishing, LLC.

Guzdial, M. (2015). Learner-Centered Design of Computing Education: Research on Computing for Everyone. Synthesis Lectures on Human-Centered Informatics, 8(6), 1–165. `https://doi.org/10.2200/s00684ed1v01y201511hci033`

Guzdial, M. (2019, April). A new definition of Computational Thinking: It's the Friction that we want to Minimize unless it's Generative,. Computing Education Research Blog. `https://computinged.wordpress.com/2019/04/29/what-is-computational-thinking-its-the-friction-that-we-want-to-minimize/`

Hemmendinger, D. (2010). A Plea for Modesty. ACM Inroads, 1(2), 4–7. `https://doi.org/10.1145/1805724.1805725`

ISTE, & CSTA. (2011a). Computational Thinking teacher resources. `https://id.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2`

ISTE, & CSTA. (2011b). Operational Definition of Computational Thinking for K-12 Education. `https://id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2`

Juškevičienė, A., & Dagienė, V. (2018). Computational Thinking Relationship with Digital Competence. Informatics in Education, 17(2), 265–284. `https://doi.org/10.15388/infedu.2018.14`

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. Baltic Journal of Modern Computing, 4(3), 583–596.

Krauss, J., & Prottsman, K. (2016). Computational Thinking and Coding for Every Student: The Teacher's Getting-Started Guide. Corwin Press.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32–37. https://doi.org/10.1145/1929887.1929902

Lewis, C. M. (2017). Good (and Bad) Reasons to Teach All Students Computer Science. In S. B. Fee, A. M. Holland-Minkley, & T. E. Lombardi (Eds.), New Directions for Computing Education: Embedding Computing Across Disciplines (pp. 15–34). Springer International Publishing. https://doi.org/10.1007/978-3-319-54226-3_2

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2019). On Thinking and STEM Education. Journal for STEM Education Research, 2(1), 1–13. https://doi.org/10.1007/s41979-019-00014-x

Lodi, M. (2020). Introducing Computational Thinking in K-12 Education: Historical, Epistemological, Pedagogical, Cognitive, and Affective Aspects [PhD Thesis, Alma Mater Studiorum – Università di Bologna]. http://amsdottorato.unibo.it/9188/

Lodi, M. (2019). Does Studying CS Automatically Foster a Growth Mindset? Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, 147–153. https://doi.org/10.1145/3304221.3319750

Lodi, M., Martini, S., & Nardelli, E. (2017). Do we really need computational thinking? Mondo Digitale, 72. http://mondodigitale.aicanet.net/2017-5/articoli/MD72_02_abbiamo_davvero_bisogno_del_pensiero_computazionale.pdf

Martini, S. (2012). Lingua Universalis. Annali della Pubblica Istruzione, 4–5, 65–70.

Martins-Pacheco, L., von Wangenheim, C., & Alves, N. (2020). Polemics about Computational Thinking: Digital Competence in Digital Zeitgeist – Continued Search for Answers: Proceedings of the 12th International Conference on Computer Supported Education, 499–506. https://doi.org/10.5220/0009797104990506

Murphy, L., & Thomas, L. (2008). Dangers of a Fixed Mindset: Implications of Self-Theories Research for Computer Science Education. SIGCSE Bull., 40(3), 271–275. https://doi.org/10.1145/1597849.1384344

Pace, D., & Middendorf, J. (2004). Decoding the Disciplines: Helping Students Learn Disciplinary Ways of Thinking: New Directions for Teaching and Learning, Number 98. Wiley.

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

Schoenfeld, A. H. (1985). Mathematical Problem Solving. Elsevier. https://doi.org/10.1016/c2013-0-05012-8

Selby, C., & Woollard, J. (2013). Computational thinking: The developing definition [Project Report]. University of Southampton (E-prints). https://eprints.soton.ac.uk/356481/

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Stephens, M., & Kadijevich, D. M. (2020). Computational/Algorithmic Thinking. In S. Lerman (Ed.), Encyclopedia of Mathematics Education (pp. 117–123). Springer International Publishing. https://doi.org/10.1007/978-3-030-15789-0_100044

Sternberg, R. J., & Ben-Zeev, T. (Eds.). (1996). The Nature of Mathematical Thinking. Routledge.

Taub, R., Armoni, M., & Ben-Ari, M. (2012). CS Unplugged and Middle-School Students' Views, Attitudes, and Intentions Regarding CS. ACM Trans. Comput. Educ., 12(2). https://doi.org/10.1145/2160547.2160551

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies, 20(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology, 25(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2011). Research notebook: Computational thinking – What and why. The Link Magazine, 20–23.

M. Lodi is B.S., M.S., and Ph.D. in Computer Science. He is currently Post-doc research fellow and Adjunct Professor of Computer Science Education at Alma Mater Studiorum – Università di Bologna, Italy. His research interest is in Computer Science Education – especially on computational thinking with a constructivist and constructionist approach, teacher training, transfer of learning, computer science mindset, and epistemological aspects of Computer Science as a discipline. He is author of more than ten publications in conferences and journals on Computer Science Education, and a book in Italian for primary school teachers. He is actively involved in nation-wide initiatives to introduce CS in Italian K-12 curriculum. `https://lodi.ml`

Codeforces as an Educational Platform for Learning Programming in Digitalization

Mike MIRZAYANOV, Oksana PAVLOVA, Pavel MAVRIN,
Roman MELNIKOV, Andrew PLOTNIKOV, Vladimir. PARFENOV,
Andrew STANKEVICH
ITMO University, Saint-Petersburg, Russia
e-mail: mrmirzaianov@itmo.ru, onpavlova@itmo.ru, mavrin@itmo.ru, rvembox@gmail.com,
shemplo@outlook.com, parfenov@mail.ifmo.ru, stankev@gmail.com

Abstract. Digitalization imposes ways of development and influences the process of Codeforces development. Codeforces' infrastructure provides a solid open ecosystem for building a programming learning process. Functionality covers the entire process, from the Polygon system for devising problems to mashup contests and private groups on Codeforces. A beta test of the educational subsystem with a pilot educational course was launched. The paper describes all the aspects and relationships of the ecosystem, typical flows of use, examples of successful integration into educational processes in the age of digitalization.

Keywords: educational platforms, competitive programming platforms, sports programming, e-learning, Olympiads and hackathons in Informatics, education digitalization.

1. Introduction

Digitalization becomes possible when there is the development of digital infrastructures and communication standards, information security, the expansion of online learning (e-learning), access and ability to use online-services and a lot of qualified IT-specialists who can create and develop information technologies, on-line services and others. Nowadays there is a shortage of IT-specialists all over the world. Their need in Russian IT- industry is 62400 people and their need in other industries of Russia is 82600 people. Enrollment in higher education institutions for IT majors in 2024 in Russian universities should be 120 thousand applicants. To ensure this number, it is necessary to train, motivate and attract more schoolchildren interested in IT and reform educational system, making it possible to teach a vast number of students.

2. Codeforces as a Platform for Programming Contests

In the world there are several mostly used contest's platforms such as American Top-coder, Indian CodeChef, Japanese AtCoder, European CSAcademy and Russian Code-forces. Codeforces is an international platform that hosts the largest regular Internet programming and Informatics competitions (Olympiads and hackathons), posts articles in IT-field, organizes programming trainings, discusses competitions & training tasks, and various news from IT-community. This is the largest training resource in the world, which provides an open infrastructure for preparing and conducting programming competitions, as well as automation of programming training courses.

Codeforces provides a wide range of services for those who are interested in algorithmic tasks and programming competitions. The main goal of Codeforces as a platform for competitions is to provide an opportunity to unite all those interested in programming contests, providing a wide list of services for this area of interest. The platform supports Russian and English languages and most of the materials are presented in two languages.

For many years Codeforces has provided three services:

- Social network (with various classic social network services and some specialized solutions).
- Subsystem of the competitions.
- Subsystem of hosting trainings.

Since December 2019 Codeforces has become the educational platform as well.

2.1. *Testlib*

Testlib library for C++ has been developed since 2005 as a replacement for an outdated similar library for the Pascal language. Currently, Testlib for C++ includes significantly more functionality compared to Testlib for Pascal. The new library has become the de facto standard for developing C++ tasks and programming tasks.
The library is widely used in the jury's work of various Olympiads and competitions:

- All the stages of All-Russian Olympiad of schoolchildren in Informatics (computer science).
- All the stages of All-Russian Team Olympiad of schoolchildren in programming.
- Repeatedly used in the development of tasks for the International Olympiad in Informatics (IOI).
- Dozens of regional competitions of the world student programming championship (ICPC).
- Many University competitions in Russia and abroad.
- Most stages of training camps (schools) in Russia and abroad.
- All rounds of open Codeforces competitions.

2.2. *Polygon System*

Polygon is a system for developing problems (task). Work on the Polygon system was started in 2008. Since March 2009, the system has been available online for everyone. Polygon simplifies and unifies the work of authors of programming problems and jury members of computer science and programming Olympiads. This is the only system of its kind with advanced functionality and accessibility to a wide range of problem authors.

Key features of Polygon that are an advantage of using it over other ways of developing problems are below:

- The Polygon system protects against errors.
- A lot of automation and self-checking tools are built into the Polygon:
  - The system protects against typos in tests from the condition and from the fact that they are not updated after changing the tests, since tests from the condition are inserted automatically, and responses to them are generated by the system by the author's decision.
  - All source texts in the problem archive (solutions, generators, checker, validator, etc.) will be compiled and correspond to the current versions.
  - The system displays a warning that the first test is not a test from the condition.
  - If the system checks that the non-deterministic generator (which is initialized from the current system time) is not being used, it will run the generator twice at intervals of a second and make sure that the tests match.
- The Polygon system implements dozens of similar checks that effectively prevents errors or ignoring good practices in problem development.
- Archives (packages) of Polygon tasks are uniform and machine-readable.
- Polygon provides long-term storage and availability of issues and competitions.
- Polygon reduces the threshold for entering the task preparation process.
- The system helps you manage access.
- Polygon has built-in issue-tracking.
- Polygon provides easy integration with automated testing systems.
- There are no special software requirements for the task developer.
- Polygon is attentive to security issues and data leaks.
- Polygon has built-in tools for classifying, indexing, and searching for issues.

According to Google Analytics, Codeforces is ahead of its competitors in terms of the number of competitions per month, the number of participants. Codeforces has an open registration system. The number of registered users is constantly growing. At the end of 2019, the main metrics according to web analytics system "Google Analytics" were:

- 472 158 users.
- 29 138 871 page views.
- Average session duration: 15 minutes.

Judging by the metrics Codeforces has a huge potential for creating on its basis an educational subsystem for participants of Olympiad programming.

3. Codeforces as a Platform for Education

3.1. *The Concept of Educational Platforms*

The debate rages in impact and effectiveness of e-learning, its benefits and drawbacks, motivations, performance and barriers (Alias *et al.*, 2012), (Zakariah, *et al.*, 2012), (Khan *et al.*, 2019), (Shapiro *et al.*, 2017), (Al-Rahm *et al.*, 2015) (Shoufan, 2019), (Shin *et al.*, 2019), (Magalhães *et al.*, 2020). This topic is becoming very actual in the time of Covid-19 pandemic. The significance of e-learning possibilities has changed and grown.

3.2. *Existing Educational Solutions*

Nowadays many online learning resources exist all over the world – Coursera, edX, YouTube, Udemy, Khan Academy. But only some of them have courses on competitive programming. The main metrics are presented in Table 1.

However existing educational platforms such as Coursera, Stepik, Universarium, EdX and others do not offer programming competitions. Moreover, the courses presented by these platforms do not meet the requirements of courses on Olympiad programming.

To conduct a comparative analysis of existing courses, the most closely related courses were selected, which are also considered the most popular in IT-community:

- The course "Sports programming" is available on the Stepik and Coursera platforms. It is the closest course in the subject
  `https://ru.coursera.org/learn/sportivnoe-programmirovanie`
- The course "How to Win Coding Competition: Secrets of Champions" on the Edx platform. It is a similar course in English `https://www.edx.org/course/how-to-win-coding-competitions-secrets-of-champions-4`
- The course "Algorithms and data structures" on the Stepik platform `https://stepik.org/course/63`

The comparison is shown below in Table 2.

Table 1

The main metrics of similar courses.

Platform	Stepik & Coursera	Edx	Stepik
Name of the course	Sports programming	How to Win Coding Competitions: Secrets of Champions	Algorithms and data structures
Metrics			
registed	2430	78015	16286
Issued certificates	129	191	542

Table 2

Comparison of algorithms and data structure courses

Platform	Stepik & Coursera	Edx	Stepik
Name of the course	"Sports programming"	"How to Win Coding Competitions: Secrets of Champions"	"Algorithms and data structures"
Course contents (main topics that should be covered in competitive programming)			
Bactracking	+		
Stack, queue		+	
Segment tree			+
Union-Find			+
Greedy algorithms	+		
Dynamic programming	+	+	
String algorithms			
DFS, Topological sorting		+	+
Shortest paths		+	+
Binary search		+	
Binary climbing			
Graph games			
Combinatorics			
Bitmasks	+		
Numbers theory algorithms			
Network flows			
Matchings			
Number and level of practical tasks in the form of problems			
Number of practical tasks in the form of problems on programming for each topic	not presented	on the average 10 problems for each topic	on the average 5 problems for each topic
Level of offered task in the form of problems on programming	not presented	mainly training problems + one-two problems similar to Olympiad problems	almost all the problems are training ones
Possibility to choose separate lectures, topics and problems	not presented	not presented	not presented
Access to the solutions of participants	not presented	not presented	not presented
Possibility to write comments	not presented	not presented	not presented
Availability of social networks for discussions	not presented	not presented	not presented
Possibility to get personal recommendations from teacher or coaches	not presented	not presented	not presented

Comparative analysis showed that the existing courses have a number of disadvantages, including:

- The material is presented in a difficult language, that is the material is intended for a more adult audience.

- Theoretical orientation (not practical).
- Basic (not advanced) level.
- They cover only a small part of the topics required for successful participation in computer science and programming Olympiads.

The authors of the paper claim that similar courses have three significant drawbacks:

- They are isolated from the competitors' community on third-party educational platforms.
- The presence of a community on Codeforces not only provides a database of participants interested in the course, but also creates a social environment for participant's communication, discussion, mutual assistance, etc.
- They use an insufficiently developed testing automation infrastructure. Support for programming tasks even of the largest educational platforms lags far behind the support for similar tasks on Codeforces.

3.3. *Codeforces – a Platform for Education*

Codeforces is well-known all over the world as a platform for contests, but lately Codeforces became the educational platform as well. The edu system has been running for three months. During this time three lessons were introduced to the users. Each lesson consists of 4-5 steps, each step includes:

- Lecture video of the algorithm (whiteboard or presentation).
- Lecture text notes.
- Coding video, explaining how to implement the algorithm.
- Programming tasks for practice.

These elements can be used in any order and independently as well as the lessons themselves. Everything is at the discretion of the learner – the pace, the order, the speed of the video, the number of problems.

The lessons were studied by approximately 3000 users. Three-month experience shows that e-learning is of great demand and the combination of platform for contests and education is very popular with IT-community. Especially it is popular with undergraduate students and schoolchildren – future IT-specialists. One can participate in codecups, learn and have talks with like-minded people at one place which is comfortable and convenient. 47 users left the feedback about edu system on Codeforces and it turned out to be highly positive. The feedback is presented in Table 3.

The feedback shows that users themselves like the idea of the educational project on the contest platform and edu project is in great demand among teens who are interested in hi-tech, Informatics and programming.

Table 3

How students express their praise or critics in relation to a new format

Aspect	Users' feedback
Praise from users	
	Cool! Continue to do such courses! Helps a lot.
	COOL!!! Good Job and Good Luck. Thanks for such an opportunity!!!
	This can help us to improve our programming skills and our knowledge. Good Job!
	Yeah that is right ! thanks a lot.
	Super! Thank you for such excellent courses! This helps not only beginners, but also advanced users. Excellent lectures, theory and practice! By the way, you are very cool to explain!) P.S. I would Really like more of these courses)
	Keep doing this great job! Thanks a lot!
	Interesting form of presentation. This option must clearly exist! Thank you for the quality courses!
	Good new system. It would be great to study more different strong algorithms.
	I have always dreamed of such a course, thank you very much!
	Just what was missing!!! Well done!!
	It is very cool in codeforces not only to solve problems but also to learn. It has become the main advantage of the platform and soon everything will be better
	Thank you so. Please don't stop. Your lectures are very useful and I really want to watch that more.
Critics and Recommendations from users	
	It would have been good to see solutions of other users if you solved the problem yourself.
	Thanks! The only thing I would like is open tests for detecting errors.
	I suggest increasing the scale of the lecturer by 2-2.5 times.
	It would be cool if you added moving (rewinding) through the video with the help of the keyboard (the same arrows)
	Can you make so that the video continues where it is left off after you switch to practice?
	Good lectures. The problems are interesting. It may be worth making the player's buttons bigger so that you could watch it via smartphones.
	A good course, and most importantly in demand. It is definitely worth developing further. I didn't find any special disadvantages for myself, so continue in the same way!
	When will be the continuation?

4. Conclusion

The development of a new format opens up a number of opportunities and has a huge practical significance. The combination of platform for contests and education give possibility to train and study at one place. Distant format provides teens with the possibility to learn for those who live in remote regions where there are no teachers and possibility for children and teenagers with a disability. Learning and contest participation is highly beneficial pastime especially during epidemics. Such approach can be used as a supplementary means in teaching because such platforms attract young people to competitions and in-depth programming studies, they facilitate to cover a wide range of topics and enlarge the number of learners, helping to nurture and form intellectual capital of any country.

References

Alias, N., Zakariah, Z., Ismail, N. Z., & Aziz, M. N. A. (2012). E-Learning successful elements for higher learning institution in Malaysia. *Procedia-Social and Behavioral Sciences*, 67, 484–489. DOI: 10.1016/j.sbspro.2012.11.353.

Al-Rahmi, W. M., Othman, M. S., & Yusuf, L. M. (2015). The effectiveness of using e-learning in Malaysian higher education: A case study Universiti Teknologi Malaysia. *Mediterranean Journal of Social Sciences*, 6(5), 625–625. DOI: 10.5901/mjss.2015.v6n5s2p625.

Khan, M. L. H., & Setiawan, A. (2019). The impact of E-learning on higher education perception, skills, critical thinking and satisfaction. *Journal of Physics: Conference Series*, 1375(1), 012084. DOI:10.1088/1742-6596/1375/1/012084.

Magalhães, P., Ferreira, D., Cunha, J., & Rosário, P. (2020). Online vs traditional homework: A systematic review on the benefits to students' performance. *Computers & Education*, 103869. DOI: 10.1016/j.compedu.2020.103869.

Shapiro, H. B., Lee, C. H., Roth, N. E. W., Li, K., Çetinkaya-Rundel, M., & Canelas, D. A. (2017). Understanding the massive open online course (MOOC) student experience: An examination of attitudes, motivations, and barriers. *Computers & Education*, 110, 35–50. DOI: 10.1016/j.compedu.2017.03.003.

Shin, J., Gruenberg, K., & Brock, T. (2019). A novel online platform promotes asynchronous class preparation and thought transparency. *Currents in Pharmacy Teaching and Learning*, 11(10), 1069–1076. DOI: 10.1016/j.cptl.2019.06.015.

Shoufan, A. (2019). What motivates university students to like or dislike an educational online video? A sentimental framework. *Computers & Education*, 134, 132–144. DOI:10.1016/j.compedu.2019.02.008.

Zakariah, Z., Alias, N., Aziz, M. N., & Ismail, N. Z. (2012). E-Learning awareness in a higher learning institution in Malaysia. *Procedia-Social and Behavioral Sciences*, 67, 621–625. DOI: 10.1016/j.sbspro.2012.11.368.

M.Mirzayanov is a founder and CEO of Codeforces and a teacher of the Faculty of Information Technologies and Programming (FITP) of ITMO University. In 2009 he coached ICPC World Champions team of Saratov University and in 2008 he coached ICPC NEERC Champions team of Saratov University. Since 2009 he has been the head of the jury of ICPC Southern Subregional NEERC. From 2008 to 2012 he was a jury member of final stage of Russian Olympiad in Informatics. In 2009 he founded Polygon – service to prepare programming problems and contests. In 2010 he founded Codeforces – programming contests community and regular Internet competitions. He was a chairman of the jury and the organizing committee of numerous championships held on Codeforces.

O. Pavlova is an assistant dean for university-business cooperation & students' well-being of the Faculty of Information Technologies and Programming (FITP) of ITMO University. Her main research interest and main professional focus is ways & mechanisms of higher education development, the creation of university and business ecosystem and ways of their cooperation.

P. Mavrin is a teacher of FITP of ITMO University. From 2014 to 2017, he was a member of the International Scientific Committee of the International Olympiad in Informatics (IOI). In 2016 he was a chairman of the Host Scientific Committee of the IOI. In 2018 he was a chairman of the Scientific Committee of the European Junior Olympiad in Informatics (eJOI). Since 2019 he has been a chairman of the Scientific Committee for the "British Programming Challenge" in London. He was given the President Award for success in the IOI (2003), St. Petersburg Youth Award in the field of information technology (2005), Saint Petersburg Government Award to teachers-mentors for preparation of winners and prize-winners of all-Russian Olympiads (2009, 2015, 2016, 2017, 2018).

R.Melnikov is a master student with major in Applied Maths and Informatics. He graduated with a bachelor's degree of ITMO University in 2019. Currently he is studying at FITP, working as a software engineer in Serokell OU, doing research for optimization problems in nanophotonics along with Physics department at ITMO University, assisting with Computer architecture and Functional programming courses at ITMO University.

A.Plotnikov is a master student with major in Applied Maths and Informatics. He graduated with a bachelor's degree of ITMO University in 2019. He leads the hobby group "Paradigms of programming" for 1st year undergraduate students; works as a senior programmer in the company "Omnics, Inc.". From 2015 to the present, he has been participating as a volunteer and coordinator of volunteers at the regional final "NERC" of the sports programming competition. Since 2015 he has been a teacher at the summer mathematical school "Spectrum", Kazan.

V. Parfenov is a professor and dean of the Faculty of Information Technologies and Programming of ITMO University, a member of international organizing committee of ICPC, the director of ICPC Northern Eurasia Finals. He is one of the main organizers and creators of national and international competitions in computer science and programming for students in Russia. He has contributed a lot to the formation of educational system of search and training gifted students in mathematics, physics, computer science and programming.

A. Stankevich is an associate professor at ITMO University. He is the Chief Judge at Russian Olympiad in Informatics (since 2019), St Petersburg Olympiad in Informatics (since 1999), Russian High School Team Olympiad in Informatics (since 2000), Individual Olympiad in Informatics (since 2009). He has been the head of Russian Central Methodical Committee in Informatics (since 2019). Judge at NERC (since 2003), Russian Olympiad in Informatics (2000-2018). Russian Delegation at IOI leader (2016, 2018, 2019). Projects: PCMS Judging System (pcms.itmo.ru), School Olympiads in Russia (nerc.itmo.ru/school), Summer Informatics School (lksh.ru). Awards: President Award in Education (2003), St Petersburg Youth Award in IT (2009), ICPC Silver and Gold Medals (2000, 2001), Seven times ICPC World Champions as Coach (2004, 2008, 2009, 2012, 2013, 2015, 2017), ACM ICPC Founder's Award (2008), ACM ICPC Senior Coach Award (2018), De Blasi Award (2013).

Pattern Recognition and Related Topics of Olympiad Tasks

Pavel S. PANKOV, Azret A. KENZHALIEV

Institute of Mathematics, Kyrgyzstan
Korea Advanced Institute of Science and Technology (KAIST).
e-mail: pps5050@mail.ru, azret.kenzhaliev@gmail.com

Abstract. In most of tasks proposed for olympiads in informatics inital data are taken arbitrary from any ranges such that a "brute force" solution (close to immediate translation of condition of the task into an algorithmical one) can pass tests with initial data from narrow ranges, and the task itself is to improve that solution. We make a survey of types of tasks where inital data are too vast to use a "brute force" but they are announced to be connected and the task itself is to extract necessary information in an optimal way. We consider tasks on recognition and on restoration of data from a unified point of view. We also make an attempt to describe various types of tasks formally.

Keywords: Olympiad, informatics, tasks on recognition, reactive tasks.

Introduction

In most of tasks proposed for olympiads in informatics inital data are taken arbitrary from any ranges such that a "brute force" solution (close to immediate translation of condition of the task into an algorithmical one) can pass tests with initial data from narrow ranges, and the task itself is to improve that solution. We make a survey of types of tasks where inital data are too vast to use a "brute force" but they are announced to be connected because they are images of certain virtual objects. The task itself is to extract necessary information about these objects in an relatively optimal way. We consider tasks on recognition and on restoration of data from a unified point of view. We also make an attempt to describe various types of tasks formally.

1. Definitions

A common reactive task can be presented as follows. All sets are meant to be finite.

Definition 1. Two computer-presentable sets (large) X and Y and a computable function $F : X \to Y$ are described (preferably, verbally). Write a program $P : X \to Y$ which imple-

ments the function F during a prescribed time (traditionally 1 second), also with restriction on memory used. Total amount of points is announced. Usually, some subtasks are given too in the following form: some sets $X_1 \subset X_2 \subset \ldots \subset X_k \subset X$ are described and amounts of points (in increasing order) for corresponding programs $P_j : X_j \to Y$ are also announced.

If the function F is defined by means of any virtual objects of a set U not belonging to X and X itself contains images of these objects we propose the following construction for some types of tasks:

Definition 2. Three computer-presentable sets (large) U, (large) X and Y, a computable function $G : U \to Y$ ("extraction of information") and a computable dyadic predicate $Q : U \times X \to \{true, false\}$ are described.

Conditions of correctness:

(*) Either $(\forall x \in X)(\exists u \in U)Q(u, x)$ or output "*No solution*" is also permitted.
(**) $(Q(u_1, x_1) \wedge Q(u_2, x_2) \wedge (G(u_1) \neq G(u_2))) \Rightarrow (x_1 \neq x_2)$.

Write a program $P : X \to Y$ such that $Q(u, x) \Rightarrow (G(u) = P(x))$, working during a prescribed time, also with restriction on memory used. Total amount of points is announced.

Some subtasks are given too in the following form: some sets $U_1 \subset U_2 \subset \ldots \subset U_k \subset U$ (also, $Y_1 \subset Y_2 \subset \ldots \subset Y_k \subset Y$) are described and amounts of points (in increasing order) for corresponding programs $P_j : X \to Y$ (or $P_j : X \to Y_j$) are also announced.

The sense of such tasks is that the evident algorithm

$$P_0(x) := (\textit{for all } u \in U) \ (\textit{if } Q(u, x) \textit{ then output } G(u))$$

is obviously too slow.

Remark 1. Restoration of u by x is impossible sometimes while value of some function (G) of u can be found by x.

The simplest example is the following:

Task 1. Let U be the set of binary 2×2 -matrices $u = \{u_{ij} : i, j = 1, 2\}$. Given four sums $S_4(u) := \{u_{11} + u_{12}; u_{21} + u_{22}; u_{11} + u_{21}; u_{12} + u_{22}\}$, found $G(u) := abs(det(u))$.
Here X is the set of tuples of four integer numbers $x = \{x_1; x_2; x_3; x_4\}$ less than 3;

$$Q(u, x) := (u_{11} + u_{12} = x_1) \wedge (u_{21} + u_{22} = x_2) \wedge (u_{11} + u_{21} = x_3) \wedge (u_{12} + u_{22} = x_4).$$

Consider two elements of $U : u_1 = \{\delta_{ij} : i, j = 1, 2\}$ and $u_2 = \{1 - \delta_{ij} : i, j = 1, 2\}$ where δ_{ij} is the Kronecker symbol. As $S_4(u_1) = S_4(u_2)$, u cannot be found. Nevertheless $G(u) = 1$ can be found uniquely.

Remark 2. From the standpoint of Definitions 1 and 2 all tasks are divided into "common" (Y is as large as X), "classification" (Y is small) and "alternative" ($Y = \{YES, NO\}$).

2. Examples of Tasks on Extraction of Information

The following task is the utterly simplified Task Character Recognition, IOI-1997.

Task 2. The file FONT.DAT contains of 26 ideal character images (a b c d e f g h i j k l m n o p q r s t u v w x y z) as 20×20 binary arrays. "Corruption" is an inverting of no more than 50 of signs in an image.

Write a program that restores corrupted images.
   Input: a 20×20 binary array being one of 26 images corrupted.
   Output: one of a b c d e f g h i j k l m n o p q r s t u v w x y z.
   The set U contains 26 elements $\{u_1, ..., u_{26}\}$.

Condition (**): the Hamming distance *H-dist* between each two different ideal character images is greater than 100.

   (The mentioned task permits some kinds of "Corruption" which seem difficult to be expressed by any "distance").
   Develop a task due to Definition 2:

Task 2a. … Input: a 40×40 binary array being one of 26 images corrupted and shifted …
   (easy)

Task 2b. … Input: a 40×40 binary array being one of 26 images shifted and corrupted …
   (difficult)

Condition (**): the *H-dist* between each shifts into 40×40 binary array of two different ideal character images is greater than 100.

   By Definition 2: The set U consists of all shifts of 26 elements $\{u_1, ... , u_{26}\}$ into 40×40 binary array. The set X contains 40×40 binary arrays;
   $X := \{x : (\exists u \in U)(H\text{-}dist(u, x) \le 50)\}$.

   $Q(u, x) :=$ "$H\text{-}dist(u, x) \le 50$". The function G transforms an image to a letter.

   The following simple task (the Kyrgyzstan quarterfinal of the ICPC administered by the ACM, Bishkek, November 2015):

Task 3. Given a sequence of four triples of natural numbers less than 100 (four distinct integer points) in space presenting a broken line of three unit links and the following flat broken lines denoted with letters:

I   L   U   Z
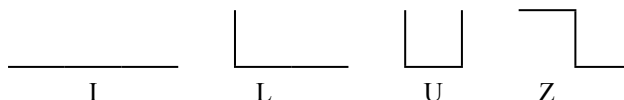
If a given broken line can be superposed with one of these then output one the corresponding letter, else output Unknown.

This task has many solutions (calculate the square of distance between the first and the last points …; calculate overall dimensions of four points …; compare the second vector-link with the first one …) nevertheless all they demand any resourcefulness and cannot be reduced to "brute force".

This task demonstrates the following features.

Given points are connected (the differences of the second and the first etc. have the length one) a proiri.

The estimation "less than 100" is not sufficient in order to search for a solution in contrast to other tasks where such estimations define their difficulty.

The set of such sequences is subdivided into five subsets (five elements of *Y*) defined non-formally.

We found an only mention of tomography in Olympiad tasks:

Task 4. Problem Giza (FARIO 2007), a brief description after (Burton *et al*., 2008).

"This task is a discrete 2-dimensional version of the general tomography problem." To recover a secret image formed from grid cells, given only the cell counts in each of 10 rows, 10 columns and 19 diagonals parallel to the main one.

Task 5 (National Olympiad in Kyrgyzstan, March 2019).

Let us call the letter T a symmetrical figure consisting of two contiguous narrow strips (width is not less than 3, height is not less than 2).

In a 8 × 9 binary array one, two or three (non-touching) letters T were depicted. The numbers of '1' bits in each of the 8 vertical columns [v-sums] and in each of the 9 horizontal lines (bottom to top) [h-sums] were detected using X-rays. For given data, find the number N of letters T and the width and height of each letter. If this is not possible, output zero.

Input: two lines of words of decimal digits; the length of the first is 8, the length of the second is 9.

Output: In the first line: the number N in 0 … 3.

If N > 0, then in each of the next N lines: two natural numbers (width and height) of the letter T, separated by a space. If N > 1 then output the data in lexicographical order: by greater width; by greater height (if there are the same, then output each one).

Example (pulled in line). Input: 01141261; 011112415 Output: 2; 5 4; 3 6

The peculiarity of this task (as well as Task 1) is that two different arrangements can yield a same input, for instance: 01411410; 111301113. Nevertheless, the answer is unique: 2; 3 4; 3 4

By Definition 2: The set *U* consists of all possible positions of 1 ... 3 "T"s on 8×9 binary array. The set *X* contains tuples of 8- and 9- words of digits; $Q(u, x) :=$ "*x contains v-sums and h-sums of u*". $X := \{x: Q(u, x)\}$.

The function *G* transforms an image to the list of "T"s that generated it.

Beginning of a possible algorithm for Task 5.

Subdivide (mentally) each "T" into the horizontal *cap* (its width is odd and not less than 3) and the vertical *leg* (height is not less than 1).

Lemma. There are no traces of three T in an h-sum (because "8" is too narrow).

Applying Algorithm-5 "If (h-sum = 5) or ((h-sum = 6) and (next h-sum > 0)) then there is a 5-cap in this line otherwise there is not" to all lines we distinguish all 5-caps.

Applying Algorithm-3 "If (h-sum = 6) then there are two 3-caps; if ((h-sum = 3 or 4) then there is one 3-cap in this line otherwise there is not" to the rest of lines we distinguish all 3-caps.

Sketch of proof of (**). Within the initial data, there are the numbers of 3-caps and 5-caps and at most three lines for them. Changing lengths of legs causes changings in v-sums necessarily.

(The jury is to write down such a proof. The contestant thinks it swiftly, together with writing a program).

Ending of the algorithm for Task 5.

As there are only a few possible positions for caps, a "brute force" search for lengths of legs from each of such positions ends the solution.

Task 6 (National Olympiad in Kyrgyzstan, March 2019, improved).

By integer numbers A, B, C, D in 0 ... 9, for the function $F(X) = ((AX + B)X + C)X + D$ the numbers $F(1)$, $F(2)$, $F(3)$, $F(4)$ were calculated and one of them was changed by less than 3. By these four numbers find $F(5)$.

3. Types of Metrical and Kinematic Spaces to Be Used in Tasks

Many of Olympiad tasks involve metrics (or distances between objects as its equivalent) or motion, evidently or latently. By our experience, changing Euclidean space to any other type of space (Borubaev *et al.*, 2003) can be made in many tasks, transforms a simple task into a complicated one (although its formal complexity does not increase) and is interesting and difficult for contestants.

Usually in tasks, 2D-Euclidean space (plane) is modified by means of "teleportation" (some given pairs of points are glued). We survey more regular constructions below.

By the way, most of spaces mentioned below were patented as variations of chess.

We will consider only integer points and metrics and motion of Manhattan type.

As a base, we take Square(s) $0 \le X \le 19$, $0 \le Y \le 19$ and Cube(s) $0 \le X \le 19$, $0 \le Y \le 19$, $0 \le Z \le 19$.

2D-cylinder: glue the side X = 0 of Square with the side X = 19 of Square. For example, the points (0; 5) and (19; 5) coincide; *dist*((17; 6), (2; 5)) = 5.

Many programmers invented the following space independently but we could not find tasks in informatics with it:

2D-torus: glue the side X = 0 of Square with the side X = 19 of Square and the side Y = 0 of Square with the side Y = 19 of Square. For example, the points (5; 19) and (5; 0) coincide; *dist*((17; 16), (2; 3)) = 10.

2D-Moebius band: glue the side X = 0 of Square with the reversed side X = 19 of Square as follows: (0; 0) coincides with (19; 19); (0; 1) coincides with (19; 18) etc.

2D-Riemann surface of square root:

Task 7 (National Olympiad in Kyrgyzstan, March 2019).

Wizard put $Square_1$ and $Square_2$ together, cut section on both ones from the center till the point $(9.5; 0)$ and glued the left side of the section of $Square_1$ with the right side of the section of $Square_2$ and the right side of the section of $Square_1$ with the left side of the section of $Square_2$. For example, $dist((9; 5) - 1, (10; 5) - 2) = 1$…. Find the distance between given two points on these Squares.

3D-torus is made of Cube by means of gluing all opposite facets.

There also exists the 3D-space "Multistory" of some floors $(Cube_1, Cube_2, ..., Cube_k)$ connected by a (prompt) elevator $(Cube_0)$ only with $Cube$-$Door_0$ $(X = 0)$ matching with all $Cube$-$Door_j$ $j = 1 ... k$ $(X = 19)$.

Metrics in this space is presented as follows. If two points P_1 and P_2 belong to a same $Cube_j$ then $dist(P_1, P_2)$ is defined as usually; if $P_1 \in Cube_0$ and $P_2 \in Cube_j, j > 0$, then $dist(P_1, P_2)$ is defined in the gluing them; if $P_1 \in Cube_i$ and $P_2 \in Cube_j$ $(0 < i < j \leq k)$ then

$$dist(P_1, P_2) := min\{dist(P_1, P) + dist(P, P_2) : P \in Door\}.$$

3. Ways to Generate Tasks on Extraction of Information

In this section we shall not consider the full procedure of creating a task. It was considered in details in Kemkes *et al.* (2007), Diks *et al.* (2008), Burton *et al.* (2008) and other publications. We propose some schemas being generations of examples in Section 2.

Remark 3. Each such task can be formulated in two ways: with "structured" and "non-structured" initial data. For instance, a complicated version of Task 3:

Task 8. Given a set of four triples of natural numbers less than 100 (four distinct integer points) in space such that after some permutation they form a broken line of three unit links …

(A proper solution is not necessarily to find such permutation: the method "calculate overall dimensions of four points …" can be applied without permutation).

Remark 4. Relatively simple tasks in multidimensional spaces are preferred because they demand imagination and practically exclude "exhaustive search".

The following task generates Task 2 and Task 6.

General task 9. There are defined a metric *dist* and a family of k connected subsets $V_1, ..., V_k$ of the set X. It is guaranteed that $dist(V_i, V_j) > 2M\delta_{ij}, M > 0$. Given $x \in X$, find $i \in 1 ... k$: $dist(V_i, x) \leq M$.

Idea of solution: using connectivity, organize effective search of such $v_j \in V_j$ that $dist(v_i, x)$ is as small as possible.

The idea of detecting compact objects by their integral indexes was proposed by A.N. Tikhonov in 1943 for detecting ore bodies underground and further was developed for medicine (tomography). We adapt it to "integer data" and, due to Remark 1, change detection of object itself to detection of some its indexes, as in Task 5.

General task 10. Withih frames of Definition 2, elements $u \in U$ are presented as sets of integer numbers. Elements $x \in X$ consist of certian sums of these numbers for all $u \in U$. Conditions on U are such that (**) takes place.

Example of task of Yes/No type where given information yields an effective solution.

Task 11. Given a sequence of $K \in 4 \ldots 10^4$ pairs of even integer numbers presenting a non-self-crossing circular broken line of links of length 2 and a pair of odd integer numbers presenting a point on a plane. Is this point within the domain bounded with this broken line? [It cannot be on the line].

The first step of the solution is detecting overall dimensions of the domain ($O(K)$ operations).

The main constituent of any algorithm of the solution is detecting whether given "odd" segment of length 2 crosses the broken line ($O(K)$ operations).

Solution 1 is the common search in depth for the growing graph of "odd" segments of length 2 beginning from the given point until either the growth stops or the graph reaches one of the overall dimensions ($O(K^4)$ operations).

But Solution 2 is the following: move along a straight line to one of the overall dimensions and count a crossing of the broken line ($O(K^2)$ operations). If the total number of crossings is odd then the point is within the domain else it is without.

General interactive task 12. A class of images is described and a procedure for queries is accessible. There is a hidden image in this class. By sequence of queries (or of limited number of queries) detect the image.

This general task also splits into two: all images are transformations of a "basic" one; images are sufficiently different.

Task 13. Given a finite set S of integer points on a plane (in a space, in 4D-space…).
  Do they belong to one straight line? Do they belong to two straight lines?
      Possible solution.
  Denote $D[I, J] := \{Z \in S: points\ Z, ZI, ZJ\ are\ collinear\}$.

  1) Take two different points Z1 and Z2 in S.
  2) If $D[1, 2] = S$ then output "one straight line"; end
     else
  3) Let $Z3$ be no collinear with Z1 and Z2.
  4) If $D[1, 2] \cup D[1, 3] = S\ or\ D[1, 2] \cup D[2, 3] = S\ or\ D[1, 3] \cup D[2, 3] = S$ then output "two straight lines"; end

     else output "NO"; end.
  Mention two popular tasks:

General task 14. Given a word of brackets (or: brackets and square brackets). Is such arrangement of brackets correct (can be obtained from any correct arithmetical expression)?

General Task 15. Given an expression containing digits, signs "+", "−", brackets. Is it written correctly? If it is written correctly then what is its value?

4. Conclusion

We hope that successful application of mathematical schemes and "general tasks" proposed above would clarify distinctions between warious types of tasks yield new tasks with "short and elegant formulation" (Dagienė *et al.*, 2007), and being interesting to solve. This would enlarge the scope of tasks involved into olympiads in informatics.

References

Borubaev, A.A., Pankov, P.S., Chekeev, A.A. (2003). *Spaces Uniformed by Coverings*. Hungarian-Kyrgyz Friendship Society, Budapest (*Chapter 4. Constructive and Computer Presentations of Uniform Spaces*).

Burton, B.A. Heron, M. (2008). Creating Informatics Olympiad Tasks: Exploring the Black Art. *Olympiads in Informatics: Country Experiences and Developments*, 2, 16–36.

Dagienė, V., Skupienė, J. (2007). Contests in programming: quarter century of Lithuanian experience. *Olympiads in Informatics: Country Experiences and Developments*, 1, 37–49.

Diks, K., Kubica, M., Radoszewski, J., Stencel, K. (2008). A proposal for a task preparation process. *Olympiads in Informatics: Tasks and Training*, 2, 64–74.

Kemkes, G., Cormack, G., Munro, I., Vasiga, T. (2007). New task types at the Canadian computing competition. *Olympiads in Informatics: Country Experiences and Developments*, 1, 79–89.

P.S. Pankov (1950), doctor of 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, 2018–2019. Graduated from the Kyrgyz State University in 1969, is a head of laboratory of Institute of mathematics of KR NAS.

A.A. Kenzhaliev (1999). Bronze medal at IOI'2016. Student of Korea Advanced Institute of Science and Technology (KAIST).

Top 10 Key Skills in Olympiad in Informatics

Marina S. TSVETKOVA[1] , Vladimir M. KIRYUKHIN[2]

[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. This article presents the methodological experience of 30 years of work on the preparation of International Olympiad in Informatics (IOI) medalists from Russia, based on the great contribution of the IOI community to the development of the syllabus IOI, criteria for the success of students and the collection of IOI tasks for 30 years of its implementation. Based on the experience of the authors of the article on the preparation in Russia of gold medalists IOI and the curriculum of school informatics are presented 10 IOI key skills as criteria for the medal success of students at IOI.

Keywords: school informatics, International Olympiad in Informatics, computer science curriculum, Olympiad training, Olympiad skills, General school competences.

1. Introduction

The support of talents in Informatics is an important vector of the development of the education system in the 21st century, which is the beginning of the digital era, the digitalization of professions, when Informatics becomes the third literacy in the education of children and requires scientific and methodological study by the efforts of all IOI countries. This is so, since IOI in 1989 was start with the support of UNESCO as a main point for the promotion of school Informatics in all educational systems of the world as a fundamental knowledge.

Over the 30 years of its development, IOI has made a huge contribution to the development of the curriculum of school Informatics, preserving the fundamental core of the subject as computer science. The formation of syllabus IOI in the 90s allowed to form a modern view of computer science in school as a science-intensive discipline, did not allow to rebuild school computer science as a course with computer applications. Currently, there is a digitalization of professions, more and more highly qualified it professionals are required, but the core of their training is the computer science. And the development of the school course of Informatics with the preservation of the fundamental core – as its basis – has become the norm for the education systems of the world. But

it requires highly qualified teachers at school (Kiryukhin and Tsvetkova, 2010), and also because of the role of computer science as a literacy program-mandatory inclusion of the course continuously from primary school. This will create an accessible environment for the formation of common competencies for all children of the country in step with the times, as well as provide an opportunity to integrate into any training and vocational education digital component at the current level.

2. The Role of IOI in the Development of School Informatics Course for the World

The role of IOI and was manifested in the methodological direction: creating an accessible Olympic Lift in Informatics (Kiryukhin and Tsvetkova, 2014), for support of talented children on Informatics in countries. Since the establishment of the IOI conference and the formation of the open scientific electronic library Olympiad of Informatics in the form of a scientific journal IOI allows all countries to focus on the experience of other, not to waste time on opening a methodological path taken already by other countries, and actively transform the existing collective expertise in each country, taking into account the specifics of educational systems in them. It is the availability of scientific and methodological achievements in the IOI environment that allows countries to focus on the level of achievements that the country needs, not only on individual achievements in the medal series of the highest dignity, but in the training system, that:

- Allows the country to form an annual team of 4 members.
- Show good result by all team members.
- To have leaders in the team with a Medal result.
- To go to the best students of the school Olympiad on professional competitions of programmers from universities, and then from the leading it companies.

This Olympic Lift is the actual engine of the development of school programs in Infiormatics in all countries of the world, IOI has become a real environment for the career growth of talented computer scientists and forms a global human potential for the development of digital civilization, and the contribution of IOI is invaluable and leading in this career lift for hundreds of IOI participants who have become an international top community of professionals ensuring the success of it companies. For 30 years, more than 6,000 children from countries from all continents of the world have received the highest experience of preparation and participation in IOI, the community of organizers has launched the Olympiad movement in Informatics in their countries, starting from primary school, motivating children in the it field.

In addition, the IOI community of trainers from countries around the world has collectively formed the content. IOI is the best in the world and is the benchmark for national computer science Olympiads, being an open live textbook for children and mentors.

Now IOI is already a world leader in the development of accessible sports environment for the children of the world, opening content tasks, including its open cloud sys-

tem events, projecting IOI experience in the world, which opened the Internet contest as training for the IOI, creating a continental Olympics for the staging controversial practices children and engages in Olympiad movement more talented kids around the world! The contribution of IOI is also the contribution of each participating country, which directs forces within the country to the development of school Informatics and is included in the partnership with countries for various events: summer camps, distance training, internships for coaches and mentors of Olympiad Informatics.

All these achievements of IOI show that IOI is a global educational and social movement of children, coaches, scientists and mentors in the field of it, which promotes the international community of scientists and methodologists, teachers and specialists of educational systems of the world, IOI remains out of politics, demonstrates the highest level of political correctness and has established itself as a public scientific movement, where the countries conducting IOI-provide a serious financial contribution to its implementation and dissemination of new achievements in Informatics.

Of course, each country has its own specifics of the education system, but it is IOI that unites countries to identify common scientific approaches to identify new methods of Olympiad Informatics and systematic approaches to the development of talents in the field of Informatics.

3. IOI Skills as Important Life Skills in the Professional Future of the Kids

The school Informatics course as IT-literacy is the basis for the Olympic lift (Kiryukhin, 2010). The sooner a child gets access to school Informatics the sooner he will be able to choose for himself the Olympic track . The basis of the Olympic lift is Olympic syllabus (Kiryukhin, 2007), it forms the profile skills, and is the Foundation of intellectual training. It is important that the first start-up in Informatics was given to every children by school, a primary school course in Informatics in 6–12 years old (Tsvetkova, 2016), motivating children in the subject, forming primary informatics literacy, based on the formation of the child's algorithmic, logical, analytical, critical, modeling, heuristic thinking, computer and computational skills, and structural memory, involving in children's competitions in Informatics in School. The development of these types of thinking and structural memory can provide Olympiad Informatics, the Foundation of which is the Syllabus IOI as profile skills in informatics literacy.

Thus it is necessary to allocate three stages of development of children in the Olympic Informatics: start-up for younger 6–12 years old, basic level for juniors 13–15 years old and advanced level for children 15–18 years old, organically to build preparation according to age and needs of children.

For each stage, a development benchmark is proposed – the development horizon, as the highest bar of achievement for each age. And the IOI community has formed all three landmarks: start up tasks (Kiryukhin and Tsvetkova, 2016), Olympiad Beavers, Junior EJOI and profile level IOI. Many National, local and regional Olympiads have become stations to test the potential of Olympians in their preparation for IOI as the world's most important Informatics Olympiad for schoolchildren.

The Olympiad track will become a portal to the it profession for all participants of the Olympiad in Informatics, regardless of the result. It is important for children to understand that their training and participation in IOI is their baggage, experience that gives them a high potential for further realization of themselves in the it profession. When we talk about the key skills in Olympiad Informatics, we mean first of all life experience and skills for the future of each child, his self-realization as a strong personality, ready to develop in the it sphere continuously.

Therefore, the top 10 skills of Olympiad training in computer science is the way to the adult future of children, some of which can show a medal result. The higher the independence of the child in preparation, commitment to the horizon of its development, perseverance and hard work, the higher its result IOI. In fact, 10 skills is a guarantee of success in training. It is important that children know about this, as the entire methodological base and practical environment of Olympiad Informatics with online tours and competitions of different levels is formed in the world and is open to children thanks to the community of IOI countries.

What skills need to be formed in the Olympiad training in computer science to maximize the child's professional growth and performance? Attention, all these skills are based on the motivation of children, their desire to work independently. If at the stage of start-up the main role in training is played by the teacher of Informatics and coaches, then for juniors this balance is 50%–50%, and for the senior group, the profile training coach performs a consulting function, the main work should be carried out by the student himself, and regularly, without failures in training, it is better every day, as do people of art and sports.

Consider the model categories of competences in the school education system and how to integrate children 's creative development into them. There are three categories of competences: to know, to be able and to apply (in practice, in life). Different accents in this system triad allow you to form different training techniques.

The knowledge approach relies on theoretical knowledge, analytical thinking, the development of intelligence, memory (to know). *The practical approach* relies on functionality, practical skills based on samples, encouraging productive experience (to be able). *The activity approach* is based on the development of self-reliance in the application of acquired knowledge and skills in life, encourages self-study and emphasizes creative initiatives of the child (to apply in practice, in life).

In the context of an information society based on dynamically changing knowledge, all three categories of competence in school education have received a new view of teachers. Thus, functional literacy (to know, to be able) based on knowledge and skills in industrial society has been replaced by a *paradigm developmental teaching* based on creativity and on the development of the capabilities of each child (to know, to be able and to apply in practice, in life).

The success of a person in the profession is largely laid down by school life. In developmental teaching it is necessary to reveal the creative personality, from early childhood to develop it and give an opportunity to self-realization in the zone of interests of the child. That is, the school of developmental teaching relies on the creativity

of the child, as much as possible contributes to his creative development. (Kiryukhin and Tsvetkova, 2011).

Methods of realization of creative potential of children traditionally are various competitions and Olympiads. Undoubtedly, for children motivated in informatics, one of such spheres of realization of creative potential is the national Olympiad in informatics and its highest manifestation – IOI.

It should be taken into account that in addition to general school education, the development of creativity is based on the development of the talent of the child in a certain sphere of creativity and requires the reflection of the specifics of creativity. It is possible to see that the main categories of child developmental teaching in Olympic preparation also are to know, to be able, to apply. However, in this training it is necessary to know the theoretical component of the sphere of creativity, to be able and to possess in perfection the instruments of creativity, to have practical experience of self-realization in the sphere of creativity in the public environment in open competition for recognition and search for ways of improvement. The methodologies developmental teaching these competencies vary. The trainer 's technique of Olympic preparation depends on how the trainer places emphasis in development of these competences, as their combination will provide.

Since each creative environment has its own specificity, in Olympic preparation for IOI it is necessary to take into account this specificity. In the general usual categories of competences to know, to be able, to apply independently it is possible to distinguish 10 key competences taking into account the specifics of IOI. At the same time, 10 key competences reflect the specifics of IOI for the development of creative activity and, as a result, the realization of children in IOI.

Can be considered 10 key skills IOI – steps of development, which will allow the child to become strong in IT, his important life experience relevant Informatics as a student, but this experience – a motor for future training and professional development. We understand that IOI despite the high level of complexity – learning environment for children's growth in science. And ahead – the real ore in the IT sphere, the innovative way of our planet in the digital environment of its development, where you need a lot of highly professional in IT. And that's the mission of IOI and everyone who works with kids in Informatics, from the school teacher to the coaches and the IOI science community.

Category of the general competences	Specifics of IOI	IOI key skill
To know	IOI syllabus	1/Subject-specialized competencies 2/Algorithmic competence 3/Learning competencies (the ability to learn)
To be able	IOI tasks archive	4/Practical competences (qualification) 5/Instrumental competencies (Programming tools) 6/Digital competencies
To apply independently	Experience of realization of personal potential in the Olympiad in Informatics National OI, IOI, ets	7/Technological competence 8/Communicative competence 9/Technical competence 10/Creative competencies

4. Top 10 Key Skills for the Success of Students in IOI Preparation

What you need to know, and be able to apply in their practice the child in Olympiad Informatics?

10 key skills in Olympiad in Informatics	Know, understand, think	Be able to do, plan, evaluate	Apply in their practice, creativity, analyze the results
Subject-specialized competencies	To know IOI syllabus, including, what topics and sections are included in the IOI syllabus, to understand the reflection of the IOI Syllabus in the tasks of the Olympiads, to think about the level of your mastery of the IOI Syllabus topics	To study the theory of sections of the IOI syllabus, to compare the problems of the Olympiad and the topics of the IOI Syllabus, plan your theoretical training, identify and eliminate deficiencies in theoretical training	To analyze the completeness of knowledge of the IOI Syllabus, to apply the theory comprehensively in solving the problems of the Olympiad
Algorithmic competence	To know the specifics of IOI problems, own various aspects of algorithmic approaches to the specifics of IOI problems on the example of the collection of IOI problems	To be able to solve IOI problems, to be able to analyze the problem and compare it with the IOI syllabus	Conduct analysis of tasks, to speak to the audience with the analysis, in practice to assess the complexity of the algorithm for solving the problem, subtasks
Practical competences (qualification)	Know about the Collection of IOI tasks, know about online resources where you can solve IOI tasks from the collection online	Be able to plan your daily workout using the IOI task collection as a basic workout for mastering the IOI syllabus	To cultivate the will to win, to achieve the solution of each problem by 100 points, to train for the speed of solving the problem
Learning competencies (the ability to learn)	To know about mass open online courses on Olympiad Informatics, distance learning systems on algorithms and programming,	Be able to learn online courses algorithms and programming, to develop erudition in the choice of MOOCs on the topics of Syllabus IOI	To plan in the preparation movement forward (positive dynamics of development), to cover new and more difficult courses, critically to estimate experience of continuous development, training and the growth in the Olympic preparation
Instrumental competencies (Programming tools)	Deeply know the tools of his creativity in Olympiad Informatics-know the technology of programming, understand the semantics of the programming language	The most complete knowledge of the fundamental tool of Olympiad Informatics-programming language C++ and additional programming languages (for IOI)	To develop skills and practice in different programming languages and software environments, to develop erudition in programming languages, their purpose and distinction, to follow the innovations
Digital competencies	To know the features of computer configuration, to know the requirements for memory and time limits for the program, to understand and understand the information structures and methods of their processing on the computer, taking into account its configuration	Own a computer, operating system and software without barriers, be able to freely navigate and work in an online environment, in a network based on analogues of the IOI competition system	Apply digital competencies in the workplace in any environment of computer support of the Olympiad, show motivation in their technical training, taking into account new computer and network technologies

Table continued from previous page

10 key skills in Olympiad in Informatics	Know, understand, think	Be able to do, plan, evaluate	Apply in their practice, creativity, analyze the results
Technological competence (experience of realization of personal potential in the Olympiad in Informatics)	Know about international and national local competitions: for example, EJOI, IAITI, USACO, APIO, Top Coder, etc., methods of self-assessment of training results, think about and critically evaluate their results, understand and evaluate their deficiencies in training and think about, plan ways to eliminate them	To be able to analyze the results of each participation in the competitions for completeness of the decision, the speed of the decision, to identify the results of the competition deficits in training on the topics of the IOI syllabus, to be able to design the results of performances and perform individual short-and long-term self-training plans	On experience to form technological skills of participation in the international, national, local competitions, to form psychological stability to the stressful competitive environment, to form confidence in the competitive ability
Communicative competence	Know the basics of communication technologies, information security requirements in the communication environment	Be able to work in English with texts, in communication, in online courses Be able to speak on the topic of Olympiad Informatics, formulate questions	To put into practice communicative literacy in independent work with sources, online courses, in the collective environment of partners and colleagues
Technical competence	Know the technical resources to equip the Olympiad environment in Informatics, the specifics of the information system of competitions, methods of work in these environments	Be able to develop high-speed and high-quality aspects of working with devices on the computer and with software, Be able to independently use and configure the necessary software	In practice, continuous monitoring of new versions of software, to analyze their differences and features in operation
Creative competencies	Full understanding of the structure of the Olympiad task, the features of the formulation of the problem in the task and its formalization, restrictions for subtasks, the specifics of the development of tests and debugging, methods of formation of the evaluation system	The ability to independently solve problems for a full score, the ability to independently compose Olympiad tasks and their full preparation for the tour (IOI analogues)	Practice self-development tasks (composed of tasks) based on the composition of skills: selection of algorithmic problems in IOI syllabus , essay text tasks in problem formulation, development of tests for this problem, the installation problem in the system events tour on the author's task for other students, participation in and development of the assessment system, conducting analysis tasks.

5. Indicators of Achievement of 10 Key Skills in IOI

Indicators for10 key skills in IOI	Indicator for stage-start up (7–12 years old)	Indicator for stage juniors (13–15 years old)	Indicator for stage 2-profile level (15–18 years old)
Subject-specialized competencies	Proficiency in IOI Syllabus topics based on the quality of tasks from the IOI collection by training level with achievement of a critical score: At least 30 points for each task for 2 hours per task	Proficiency in IOI Syllabus topics based on the quality of tasks from the IOI collection by training level with achievement of a critical score: At least 60 points on average for three tasks in 2 hours per task	Proficiency in IOI Syllabus topics based on the quality of tasks from the IOI collection by training level with achievement of a critical score: At least 75 points on average for three tasks in 2 hours per task
Algorithmic competence	The number of tasks from the IOI collection (performed on a critical score) per year is not less than 10	The number of tasks from the IOI collection (performed on a critical score) per year is not less than 20	The number of tasks from the IOI collection (performed on a critical score) per year is not less than 30
Practical competences (qualification in IOI)	The number of tasks solved in a year, modified to a full score – at least 10	The number of tasks solved in a year, modified to a full score – at least 20	The number of tasks solved in a year, modified to a full score – at least 25
Learning competencies (the ability to learn)	Number of online courses for this training group per year is 1–2	Number of online courses for this training group per year is 3	Number of online courses for this training group per year is 3–5
Instrumental competencies (programming tools)	Number of courses in C++ programming language at least 1	Number of courses in programming languages at least 2 Number of courses in C++ programming language at least 1	Number of courses in programming languages at least 3 Number of courses in C++ programming language at least 1
	Speed of solving IOI collection problems in C++ per critical score no more than 1 hour per 1 task		
Digital competencies	Assessment of software proficiency under Linux OS (mastered common tools in Linux OS)	Assessment ownership software under OS Linux (not fully, can work on Olympics under OS Linux)	Evaluation software ownership under Linux OS (Yes, full ownership)
Technological competence	Number of online tours or competitions-1 each month	Number of online tours or competitions-2 each month	Number of online tours or competitions-3 each month
	The average score for tours and competitions per month is not less than the critical score for the preparation group		
Communicative competence	The level of understanding of the interface of the system of competitions in English: "I Understand partially"	The level of understanding of the interface of the system of competitions in English: "I Understand, there may be questions» Level of understanding of the text of tasks in English, online courses in English (I understand partially)	The level of understanding of the interface of the system of competitions in English: "I Understand freely» Level of understanding of the text of tasks in English, online courses in English (full)

Table continued from previous page

Indicators for 10 key skills in IOI	Indicator for stage-start up (7–12 years old)	Indicator for stage juniors (13–15 years old)	Indicator for stage 2-profile level (15–18 years old)
Technical competence	The level of knowledge of the technical capabilities of the competition system: training, work with external technical support and advice of the coach	The level of ownership of the system of the competition: possess, can be issues when moving to a new system of competition takes time to adapt	The level of ownership system contest: possess freely, can on their own quickly adapt to any system contest for negligible time
Creative competencies	Participation in the task development group with a coach, experience of participation-at least 1 task per year	The Number of tasks fully developed in a group with a coach like IOI for training tours in their training team – at least 2 tasks per year	The number of tasks fully developed in a group without a coach like IOI for training tours or competitions-at least 2 tasks per year

6. Role of Head Coach for the IOI National Team's Olympic Training

Personal experience as the head coaches of the Russian national team from 1989 to 2018 allowed us to find ways to guarantee the success of children participating in IOI in practice.

First, let's list what the head coach is responsible for:

- The final result.
- The organization of the training process.
- Potential growth in the preparation of participants (together with teachers, mentors and coaching staff).
- Building a long-term plan for each stage of training (for 3 years) for each participant.
- The team's psychological readiness to perform successfully.
- Basic IOI-specific training (i.e. the minimum IOI-specific qualification level is the individual training plan of each participant and the quality of its implementation by the participant).
- Long-term planning in the framework of new trends in Olympiad Informatics.
- The team selection methods based on objective results (indicators) of the participant's preparation, taking into account the achievement of the qualification minimum according to IOI specifics.
- The formation of basic training for juniors as a shift to the senior team.

Often coaches, especially those who are unfamiliar with pedagogy and school work, focusing on the professionalism of students, do not take into account the full readiness of the participant for the specifics of the competition, believing that any competition will have an effect. This is not only wrong, but also harmful for children, since the spread in different competitions takes a lot of effort from the participant and does not allow them to prepare for medals based on the specific specifics of the target competition.

In order for participants to grow, they must first be taught to train well, and it is the Junior stage that is aimed at this task.

You need to start in a timely manner, since entering the Olympiad qualification as a training base from the age of 16 will not give the desired effect due to lack of time, since the qualification is specific to a specific Olympiad. A Junior progresses only when he goes through a well-thought-out, well-targeted training within a well-organized training process. Target training, or qualification training – is a painstaking daily work, training on the IOI tasks archive (IOI, 2019) as is customary for musicians, artists, and daily sport training or art etudes. The IOI archive is such mandatory etudes or daily training sessions for preparing for IOI.

There is a common misconception that participants allegedly grow during participation in various competitions. This is not true. In the competition, they only demonstrate what they have accumulated over many days during hard training. And if there is no good training process, or if the participant himself for some reason does not work well or is not motivated to succeed, then he does not show high results in the competition. Frequent participation in competitions indiscriminately, mixing of specifics in competitions without targeted training leads to fixing the participant's incomplete result on the tasks of the competition, no higher than 50 percent of points and lack of perseverance in training. As a result, the opposite effect is fixed – the inability to work for a full score, the highest result and the unwillingness to hone it in independent constant work. You need to learn how to get a high result, and most importantly, do it at the Junior stage, so that there is time to improve the participant at the peak of his capabilities, that is, to open the child to take off to his own capabilities.

Talent plus every day work is the key to the success of any participant. The presence of talent can be recognized by an expert look and the first performances in national Olympiads of children.

But, unfortunately, many gifted participants believe, or have been given the wrong installation by their personal trainers, that it is enough for them to perform on one talent, watch and listen to others, but not necessarily do something themselves. They expect enlightenment by training irregularly and not full strength.

You can achieve real success only through colossal, regular, daily work and a deep passion for the subject of the Olympiad. All those who become IOI stars work hard on themselves for a long time. We can say that the key to improving talent is the balance of coaching and independent work 50–50. If you do not teach the child to organize their own regular work to improve their talent, they will show the same 50 percent of the result. And if at the Junior stage the coach's participation prevails, then at the Junior stage this balance must be clearly carried out, then at the stage of the senior group motivated independent work prevails and this is the key to full results.

Take examples of IOI participants who were at the peak for 3–5 years, at the top of IOI, who became absolute Champions twice or more times. Behind these achievements is a daily, passionate, correct training work. In teams where the medal result of all team members in each IOI year is shown, such work is built. It is important to analyze the statistics of such achievements of countries and take into account their experience.

Concentration of attention, discipline of the training mode, high motivation, deep training, independence and perseverance-all this should be brought up in the participant from an early age and taken into the work of the head coach, then you can plan the growth of the success of talented children to the level of their potential opportunities for everyone, and this should be a guideline for the medal result. With these opportunities and important experience, children will go further into professional life.

7. Methodology of the Basic Qualification Olympiad Preparation of the IOI National Team

How to plan regular, daily work-basic (qualification) training of the Olympiad preparation? The difference between the basic qualifications is that it is an invariant core of training that is common to all participants, in addition to its variable individual trainings that take into account the child's personality and potential.

Basic training as a qualification, mandatory for all participants in preparation for the international Olympiad, can be represented by a three-level system.

The training is conducted on the tasks of the archive of the Olympiad that the participant is preparing for. We will look at the IOI goal. Basic training for IOI is conducted in two-week cycles, each cycle includes thematic training sessions and tours on IOI tasks on the topic.

The basic training curriculum for each age group of training is designed based on the IOI syllabus topics.

Cycle tours have different specifics, but are intended for all age groups: primary group 7–12 years old, junior group (13–15 years old)? Senior group (16–18 years old). For all age groups:

1. **Thematic tour** with a selection of problems on a specific topic from Syllabus IOI with a preliminary lecture on this topic.
2. **A validating tour** for knowledge of the specifics of tasks from different topics, using a ready-made tour of the year from the Olympiad archive (IOI or at the choice of the coach from similar competitions) with a preliminary overview lecture on the topics in the tasks of the tour.
3. **A control tour**, such as a type 1 and type 2 tour with a final lecture after the tour.

Children should not be divided into classes based on age. The complexity of the task of tours is regulated for the groups, but all classes are held together.

Cycle for 2 weeks of training. For 2 hours a day, the day of the tour for the allotted number of hours is recommended to be spent on an extra-curricular day at school.

First week, 6 days.

- Days 1–2, (2 days) 4 hours. Lecture on the theory, on the topics of IOI syllabus, for which the tasks for the tour are selected from the archive of IOI (or other Olympiads).

- Day 3, (one day). Conducting a tour on 2–3 tasks for 3–4 hours, respectively. For different age groups, tasks are selected from the corresponding difficulty tour, and the achievement of a critical score for the age group is recorded. Primary group have one task for 2 hours, junior group have two tasks for 3–4 hours, and senior group have 3 tasks for 4 or 5 hours.
- Day 4 (one day), 2 hours. Selection of algorithms for solving problems of the tour. The analysis is carried out in two stages. First, the participants with the best results come forward and explain their solution. Then the trainer shows the reference solutions for the full score . only then- explanation or final lecture of the theory of Syllabus IOI on the example of a reference solution of the task.
- Day 5 (one day), 2 hours. Repeat the tour on the same tasks for 1 hour for juniors and for seniors groups. The participant must not see their decisions and repeats the tour again, but for what time. This will show the effects of absorption of the solutions at the critical point and above, speed and quality. Often this tour reveals the technical and instrumental, fault intellectual deficits of the participant's training.
- Day 6, (one day). Individual consultation with the coach. 1 hour. Filling in the individual plan by a student with a self-assessment of all indicators for the week. Assignment of 3 tasks from the Olympiad collection by the coach for independent study and trial solution in 6 days. Tasks are selected either by the passed topic (for any age group) or as a tour of the year from the archive with self-analysis of the Syllabus IOI topics of these tasks (for juniors and seniors).

It is not necessary to select separate tasks for the Primary and Junior age groups. All IOI tasks have different solution levels, separated by subtasks and provided with points. For each age group on the same tasks in the allotted time, it is necessary to achieve bringing the solution of the problem from the IOI archive to the critical score for the age group. As a result, according to the basic training (IOI qualification), each participant at the Junior and senior group level must already demonstrate the solution of problems to the full score (even with repeated approaches to the solution). It is very important that the participant has sufficient experience in solution tasks from the IOI archive for a full score and demonstrates the analysis skills these tasks and solve them again in a very short time at speed.

Group consultation with the coach. 1 hour. General discussion of tasks, identification of Syllabus topics for these tasks, and distribution of IOI tasks (subtasks to them) by age groups.

The second week, 6 days.

- Days 1–4, 10 hours. Home / independent work. Self-completion mode for the full score of the tour tasks and thinning out the selected 3 tasks from the Olympiad archive. Filling in the final table of indicators. Preparation of the analysis of 3 problems on the topics of the syllabus. The tasks analysis card includes: which topic of the Syllabus IOI is the task, what algorithm use for this task , and estimation of the complexity of the algorithm.

- Day 5, 2 hours. Seminar on the results of the week of homework. Analysis of home independent work. Presentations of students on tasks with independent analysis. Group consultation with the coach.
- Day 6, 2 hours. English language training, reading and analysis of task texts in English from the IOI tasks archive and other Olympiads with the participation of a coach. Brainstorming on approaches to solving these problems.

8. Starting and Planning Competence of the Participants of the National Olympic Reserve School

Trainings can be organized in the country in the form of winter and summer schools and distance sessions between them.

The focus-group of training is juniors (13–15 years old) include 3 training cycles within three years to achieve planned competencies based on the formation of a culture of independent work. Seniors group (16–18 years old) select at result of working with juniors after 3 years sessions, training with seniors is carried out on an individual route with a personal trainer and a serious training plan for the results of higher achievements.

The selection of successful juniors is made from the primary group. It is important to conduct for this selection a mass school stage of the national Olympiad in Informatics, where successful children of 12–13 years old are identified, and a national team of juniors is formed for training sessions for them of the national Olympic reserve school and their promotion to higher achievements.

With young students (7–12 years old), the work is carried out in primary groups in schools in their places of study with the involvement of children in online competitions. For younger students, it is important to master the course in Informatics and introduce syllabus topics based on problems in logic, combinatorics, graphs, as well as to master the programming of simple algorithms in C++ at the user level. Summer and winter sessions for primary groups are held in their schools. For this purpose, teachers of these primary students must have an Olympiad mentor, with whom they can consult at any time, including Internet communication line, and have methodological materials and sets of tasks for younger students about Olympiad training.

Training camp of the national Olympic reserve school for juniors and seniors are best conducted together.

This School for junior and senior students is held as an Olympiad Informatics camp for national teams. Age of participants is supposed 13, but to be not older than 16 years by the end of the year. The student can choose the level of training: advanced (A) or base (B). The content of classes of the school of winter and summer schools is based on the thematic sections of the Syllabus IOI "6.2 Algorithms and Complexity (AL)".

IT clubs on the use of IT in various fields of science and technology is to form a space of formation of profile interests at the junction of science and information technology. The School program provides Robot-club.

Coach-group of the national Olympic reserve school:

- Trainers of the national Olympiad in Informatics, coaches in Algorithms, IOI – tasks coaches, C++ programming teacher, specialist in the competition system.
- MOOC-tutor, consultant on new digital professions and it development prospects, General digital literacy coach or digital curator (Tsvetkova and Kiryukhin, 2019).
- Robot-club coaches (samples of application of programming in engineering).
- additional coaches in chess algorithmic games (origami, Rubik's cube, Go game, etc),
- Sports coach, educational psychologist for the junior and senior teams.

Start-competence of new participants of the national Olympic reserve school (juniors 12–14 years old)**:**

- Knowledge of the subject of Olympiad tasks at the level of the National Olympiad in Informatics, high results at the school stage Of the National Olympiad in Informatics.
- Primary experience in solving simple problems of Olympiad Informatics (national Olympiad level, or levels of simple problems in the IOI task collection). Select a simple task from the collection can be based on the statistics of its solutions to the full score, in the Statistics section on the IOI website (IOI , 2019).
- Introduction to the themes of Algorithms.
- Availability of primary experience of participation in online Olympiads in Informatics.
- Ability to use the system of competitions, to send the decision for check, to test the decision.
- Primary C++programming skills.
- Ability to speed input on the keyboard Planned personal Junior skills on ISIJ The individual plan of the Olympiad training (for the period of 2–3 years) is focused on the independent work of the Junior with the resources of the Olympiad Informatics taking into account the specific tasks of IOI and syllabus IOI.

Planned competence of participants of the national Olympic reserve school (15–16 years old)**:**

- Knowledge of the IOI Syllabus.
- Ability to analyze the problem and determine the theme of the Syllabus IOI for the tasks.
- Experience in solving problems of increased difficulty theme.
- Knowledge of the criteria of self-assessment of achievements based on solving problems of increased difficulty, the ability to conduct self-analysis on the results of training, to set tasks to eliminate the identified deficiencies.
- The use of MOOC for development of the Olympiad in Informatics, Algorithms, data Structures, Programming.
- Readiness to solve the tasks of summer and winter school for daily training the practice on a full score.

- Ability to identify difficulties in solving the problem in the first approach to it, the ability to find and study the theoretical material and analysis of solutions on the basis of the first approach to eliminate deficiencies, the ability to bring the solution of the problem after the second/subsequent approaches to a full score.
- The ability to build the personal training plan for the 2–3 years (a daily 2 – hour lesson on the tasks, in a remote environment, the ability to implement your plan without missing).
- The ability to work out high-speed problem solving skills *(Yandex. Contest,2018)* repeated solution of one problem after its solution and analysis for a full score not more than 1 hour.
- Regular participation in a MOOC to improve the skill of programming in C++ (for example Cursere , Cisco Networking Academy).
- Regular participant in online Olympiads.
- Elimination of difficulties in the English-speaking environment of the competition system and Olympiad tasks.
- The ability to analyze the text of the problem, the ability to analyze in the group solutions for subtasks and complete solutions.
- Fluency in work with systems of competitions in Informatics.
- The experience of creativity to develop algorithmic problem and legend-text of task and tests-set for the tour (work in a group with a coach), its full preparation for inclusion in the tour.
- The communication culture skills with colleagues.
- Understanding the value of active rest and sports for the programmer, compliance with the time for recreation and creativity in the application environment.
- Knowledge in perspective professions in the IT sector.

9. Conclusion

10 top skill is a key tool for solving the coaching tasks described above and a guaranteed path to the success of purposeful talented children in Olympiad in Informatics.

As leaders of the Russian team at the IOI from 1989 to 2018, we offer coaches of countries to use the methodology of Olympic training received in Russia, modified and issued as a coaching technology for the years 2002–2018. Since 2017, we have implemented in Russia the selection of the Junior team for training as an Olympic reserve for the selection of teams for IOI and opened this experience to all countries at the international school of Informatics for juniors ISIJ (ISIJ, 2019), described the starting and planned competencies of juniors to prepare for IOI. ISIJ takes place every year in Russia, Kazan in the summer, but in the winter – in countries by request.

This method was implemented in the preparation of Russian teams, the coaching goals were achieved, Russian teams always took 4 medals at all IOI and had gold. Now when preparing juniors, in addition to the strong training of the senior team, a groundwork is laid for success for several years to come: the succession of medalists. The guarantee of success in the stable preparation of the country's team with the formation

of a change of teams from juniors-gives a start for 3 years ahead, as demonstrated by the Russian team, consisting of former juniors since 2018, the result of gold medals in 2018 and in 2019 confirms the success of the methodology developed by us. All participants of the Russian team of the last years started their training as juniors from 12–13 years old using the method described above. It is important to keep the best experience, enrich it, and rely on the methodological achievements and experience of the older generation when changing team coaches in the country, which is inevitable for decades.

The result of this method is the current national system of preparation for IOI in Russia, which can be used by any IOI member country. We are open to sharing this experience, which can be implemented in the work of coaches for all IOI teams. Training in this method for Junior teams and coaching groups from the country can be held at the annual international school of Informatics ISIJ in *Russia* (Tsvetkova and Kiryukhin, 2018).

The authors want to thank the entire IOI community for this huge contribution that IOI has made over 30 years of work, and wish the new generation of team leaders-head coaches-not to lose valuable techniques created during this period of formation and development of IOI and national computer science Olympiads for schoolchildren in the world. I suggest using this experience and going forward with it, looking for new trips to maximize the creative potential of the participants of the Olympiad. The main thing is that participation in IOI on the basis of talent and hard work will become their important life competence for further creativity and professional success.

Reference

Kiryukhin, V.M. (2007). The modern contents of the Russian national olympiads in informatics. *Olympiads in Informatics*, 1, 90–104.

Kiryukhin, V., (2010) Mutual Influence of the National Educational Standard and Olympiad in Informatics Contents. *Olympiads in Informatics*, 2010, Vol. 4, 15–29.

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., Tsvetkova, M. (2011). Preparing for the IOI trough developmental teaching. *Olympiads in Informatics*, 5, 44−57.

Kiryukhin, V.M., Tsvetkova, M.S. (2014). The approach of early olympiad preparation "Olympic Lift". *Olympiads in Informatics*, 8, 111–122.

Tsvetkova, M. , Kiryukhin, V., (2016) Concept of Algorithmic Problems for Younger Students Olympiads in Informatics. *Olympiads in Informatics*, Vol. 10, Special Issue, 67–78.

Tsvetkova, M., (2016) Informatics at Russian Primary School. *Olympiads in Informatics*, 2016, Vol. 10, Special Issue, 3–6.

Tsvetkova, M. , Kiryukhin, V., (2018) International School in Informatics "Junior" for IOI Training. *Olympiads in Informatics*, Vol. 12, 187–193.

Tsvetkova, M. , Kiryukhin, V., (2019) Digital curator. *Olympiads in Informatics*, Vol. 13, 237–240.

Yandex. Contest (2018) IOI Archive. https://contest.yandex.ru/ioi/?lang=en

ISIJ (2019). International school in informatics "Junior". http://www.isi-junior.com

IOI (2019). International Olympiad in Informatics. Statistics. http://stats.ioinformatics.org/tasks/

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

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

REPORTS

Argentine Olympiad in Informatics

Agustín Santiago GUTIÉRREZ
Universidad de Buenos Aires, FCEN, Argentina
e-mail: asgutierrez@dc.uba.ar

Abstract. This article describes Argentina's selection and training process for the International Olympiad in Informatics, to identify and prepare the Argentine team members participating in the international olympiad. Additionally, recently developed online resources that have proved very useful for student preparation are presented.

Keywords: olympiads, programming contest, selection, training, grading systems.

Introduction

In the year 1990, Argentina started to participate in the International Olympiad in Informatics (IOI) (Kalinicenko, 2020). However, the Argentine Olympiad was under a different administration at that time, and it was suspended after the 1994 IOI. Argentina did not participate in the IOI 1995 and 1996 editions. Very few contacts, statistics and records are available from this period.

In 1996, OIA ("Olimpíada Informática Argentina", that is, the Argentine Olympiad in Informatics) was started once again (OIA, 2020c), now based at General San Martín National University (UNSAM) in the Buenos Aires Province. Since then, the olympiad has been running continuously at UNSAM, selecting and sending delegations to IOI every year. OIA is part of a general National Program of Olympiads (Argentina.gob.ar, 2020), in which the Ministry of Education assigns the task of organizing the different Science Olympiads in Argentina to National Universities.

Up to and excluding 2020, Argentina has participated in 28 IOIs, obtaining 3 gold medals (1990, 2003, 2018), 9 silver medals, and 23 bronze medals.

IOI Selection Process

General Contest Environment

In all of OIA contests, each task is graded from 0 to 100 points, similar to IOI.

All contest up to and including year 2013 were run either by manually collecting and testing all the contestants' code at the contest's end, or by using custom OIA Contest Management Software.

From 2014 onwards, CMS (Maggiolo and Mascellani, 2012; Maggiolo *et al*., 2014) has always been used for the National Contest and the Selection Contest. This has been an enormous improvement in term of Contest Management quality, and has also allowed students to be already familiar with the IOI Contest Environment before actually arriving at IOI.

Except for the Selection Contest (last round), only Batch-type tasks have been used so far in the olympiad.

First round: Jurisdictional Contest

The Jurisdictional Contest is open to any high-school student in Argentina. It is divided into three levels, based on school-year (counted from first grade, that is, the first year of primary education):

- Level 1: 8th and 9th grade.
- Level 2: 10th and 11th grade.
- Level 3: 12th and 13th grade.

The country is divided by OIA into *jurisdictions*, a term chosen to encompass at the same time provinces, Buenos Aires City, and regions of large provinces. Each jurisdiction has a Jurisdiction Coordinator appointed by OIA. The country is divided into 25–30 jurisdictions, with the precise number varying in recent years.

In 2019, the Jurisdictional Contest was coordinated nationally by OIA for the first time (OIA, 2020d). It is now a single contest across the whole country: students gather at schools designated by their jurisdiction's coordinator, and log into an online running CMS server in order to compete.

Previously, the jurisdiction coordinators were in charge of selecting its contestants for the National Contest however they judged appropriate, being required only to present a final list of students of at most 3 per level per jurisdiction. Since 2019, a single national contest is prepared by the national judges (aided by helping judges from the different jurisdictions) and the whole country has to compete using this same contest on the same day of competition. This has been a significant improvement in terms of statistics gathered by OIA about its student base.

The contest has a length of four hours, and consists of **four** tasks per level. It is the judges intention that the first of the four tasks in each level is very easy: it should take

experienced, well-prepared contestants in the corresponding levels no more than 5–10 minutes to read and fully solve these tasks. This decision, as well as the total of four tasks, helps in successfully using a single contest throughout the whole country, accommodating large differences in level among the different jurisdictions.

232 students submitted solutions in this round in 2019: 36 in level 1, 104 in level 2, and 92 in level 3.

Top students advance to National Contest. The previous rule of selecting always 3 students per level per jurisdiction has been changed in 2019 (OIA, 2020e; OIA, 2020f). Now, only the best **two** students (as determined by their scores in this unified round) per level per jurisdiction are selected. Based on the available budget for the National Contest, additional wildcard slots are added for the next best scoring students, regardless of their jurisdiction.

Second Round: National Contest

This is an onsite contest. All of the students are gathered in the same city, with expenses covered by OIA.

The National Contest is divided into three levels, exactly the same as the Jurisdictional Contest. In the last decade, about 60 to 100 students have partipated in the National Contest each year.

The contest has a duration of four hours, and consists of **three** tasks per level. After a single contest day, medals are awarded to the top contestants in each level.

Traditionally, exactly three medals per level were given: Gold for the best contestant, silver for the second best, and bronze for the third place. Since 2019, the medal allocation algorithm was changed based on the number N of contestants achieving at least 25% of the total possible score:

- Gold: $\frac{N}{10} = 2\frac{N}{20}$ medals.
- Silver: $3\frac{N}{20}$ medals.
- Bronze: $\frac{N}{5} = 4\frac{N}{20}$ medals.

This was done both to allow more medals in cases where many good contestants appear in the same level, and to prevent bronze medals from having extremely low scores. The number of medals in recent years would not have changed much if the rule had been in place, except when these special cases occurred. The jury is allowed to decide on the precise cutoffs in the case of ties and near ties.

The top students in the National Contest are allowed to advance to both the Selection Contest and the Iberoamerican Olympiad in Informatics (CIIC) (OEI, 2020; OIA, 2020a). As stated in OIA rules (OIA, 2020e), the advancing contestants are the top **three** students of level 1, the top **four** students of level 2, and the top **six** students of level 3. The jury is allowed to make additional students advance based on their performance in the National Contest.

Third Round: Selection Contest

The final round is the Selection Contest. The goal is for it to be as similar to the IOI contest as possible: that is why it is organized in two days of competition, having a duration of five hours and three tasks per day, and separated by a free day without competition. Traditionally, Batch-type tasks were used exclusively from 2000 to 2015, but in recent years other type of tasks have been included, in alignment with typical IOI tasks types:

- Interactive tasks: The first interactive task ever used in OIA was *Iluminando el árbol de Navidad* (Lighting up the Christmas Tree), used in 2016 (OIA, 2020b). A total of four interactive tasks have been used in the Selection Contest, exactly one per year:
  - Iluminando el árbol de Navidad (2016).
  - Encontrando el Tiranic (2017).
  - Secuenciando el ADN (2018).
  - Cultivando bacterias (2019).
- Output-Only tasks: The first output-only task ever used in OIA was *Contando Subredes* (Counting subnetworks), used in 2018 (OIA, 2020b). A second output-only task, *Recuperando distancias* (Recovering distances), was used en 2019. These remain the only two output-only tasks to have ever been used, before and excluding 2020.

The top four contestants, by total score after adding together both competition days, are selected for the IOI Team.

Number of Participants

The total number of contestants per round and year for recent editions of the olympiad is shown in Fig. 1.

Student Preparation

Dedicated Training by OIA

Once the IOI Team is selected, it is invited to attend an onsite-training, typically lasting about a week. This is an intensive, focused training event where the four students are lectured in different topics during the morning by an instructor, and are given time to solve and code homework problems during the afternoon.

Apart from the onsite training, the team is also in contact with an instructor for remote training up to the time of the IOI itself. This training consists mainly of the assignment of reading material and homework problems by the instructor, as well as asking and answering questions by email or messaging.
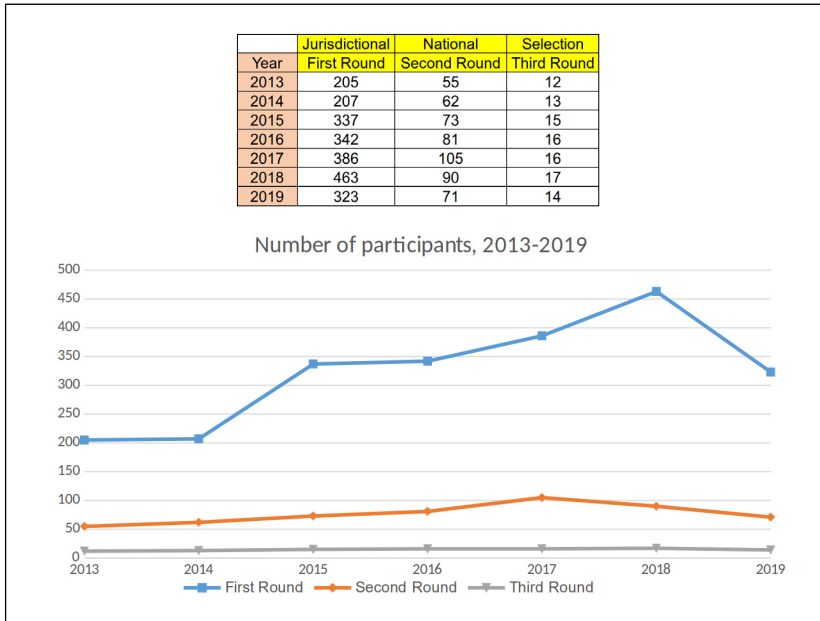
Year	Jurisdictional First Round	National Second Round	Selection Third Round
2013	205	55	12
2014	207	62	13
2015	337	73	15
2016	342	81	16
2017	386	105	16
2018	463	90	17
2019	323	71	14

Fig. 1. Number of participants (2013–2019).

OIA Online Resources

In order to help students and teachers to prepare for the olympiad, OIA has several on-line resources focused on high school informatics olympiads in Spanish. Except for the webpage itself, all of these resources were launched in 2017, and have already been very well received by teachers and students in Argentina.

- Website: `http://www.oia.unsam.edu.ar`
- YouTube Channel: (Olimpíada Informática Argentina)
  `http://www.youtube.com/channel/UChUJZDER53wT7VNT_UwCJyw`
  There is not much video material by OIA yet, but that which has already been published is available via this channel.
- Wiki: `http://wiki.oia.unsam.edu.ar`
  This is a website aimed at providing trusted, high quality reference material about programming competitions in Spanish, mainly focused on high-school level competitions like OIA. It is not an open wiki, but only OIA's instructor's and trusted ex-competitors are allowed to write articles. An important ongoing effort consists in completing and improving this site.
- Forum: `http://foro.oia.unsam.edu.ar`
  This official OIA forum has proved quite helpful in creating a sense of online community around OIA.
- Judge (OIAJ): `http://juez.oia.unsam.edu.ar`
  An online judge dedicated to hosting OIA's problems. It is directly based on cmsocial (GitHub, 2020), the direct sucesor of oii-web (Di Luigi et al., 2016)

The judge has improved student preparation enormously, as it gives access to an easy to use online judge very similar to the real CMS used during OIA's contests, with an interface and competition tasks in Spanish. Many workshops and training sessions have used OIAJ as its main online judge. Users from various countries in Latin America have used OIAJ as a source of competition problems in Spanish.

OIA Additional Resources

Booklets ("Solucionarios"): Starting since 2017, an official booklet is published containing solutions and explanations to all the problems used that year in the olympiad ($12 + 9 + 6 = 27$ problems per year). The 2017 booklet was completed and published (OIA, 2020g) in the beginning of 2019. The 2018 booklet was completed in december 2019, and is expected to be released in early 2020. The 2019 booklet is being written and is expected to be released sometime during 2020.

- Talks: Since 2014, talks have been given during the National Contest event. These are typically lectures by ex-competitors focused on some competitive programming topic. Also, a *lecture for beginners* has been given many times, and is quite well received by teachers and students. This is a lecture explaining how to solve simple problems using for loops, how to understand statements, and such fundamental skills.

  Many of these talks are filmed and have been published in OIA's Youtube Channel. Most of them have slides available in OIA's webpage (OIA, 2020h).

Future work

It is motivating to see that many of these new resources launched by OIA in recent years have been very well received by teachers and students. There is however a lot of room for improvement. Some important goals for the coming years are listed:

- Upload to OIAJ all problems having a statement published in OIA's webpage so that they can be submitted there.
- Improve the wiki so that it covers material easily available in English (C.P. Handbook (Laaksonen, 2017), cp-algorithms.com, etc)
- Increase participation in the forum.
- Make it easier to start participating. Some ideas: visual tasks, visual programming, paper-and-pencil exams.
- Increase the total number of participanting students and schools.
- Create OIA training camps, and more generally, increase the availability of training courses.

References

GitHub (2020). CMSocial. A web application that builds a social coding platform upon cms.
`https://github.com/algorithm-ninja/cmsocial`. Accessed: 2020-02-08.

Argentina.gob.ar (2020). Ministry of Education. National Olympics Program.
`https://www.argentina.gob.ar/educacion/programa-nacional-de-olimpiadas`. Accessed: 2020-01-18.

OIA* (2020a). *CIIC : Competencia Iberoamericana de Informática y Computación.*
`http://www.oia.unsam.edu.ar/ciic-reglamento`. Accessed: 2020-01-18.

OIA (2020b). Competition Tasks.
`http://www.oia.unsam.edu.ar/problemas-categoria-programacion/#seleccion`. Accessed: 2020-01-24.

OIA (2020c). History. `http://www.oia.unsam.edu.ar/antecedentes`. Accessed: 2020-01-18.

OIA (2020d). Nationally Coordinated Jurisdictional Contest. `http://www.oia.unsam.edu.ar/instancia-jurisdiccional-oia-2019`. Accessed: 2020-01-18.

OIA (2020e). Rules. `http://www.oia.unsam.edu.ar/reglamento`. Accessed: 2020-01-18.

OIA (2020f). Rules – Structure. `http://www.oia.unsam.edu.ar/estructura`. Accessed: 2020-01-18.

OIA (2020g). Solucionario OIA (problemas 2017). `http://www.oia.unsam.edu.ar/oia-programacion/`. Accessed: 2020-02-08.

OIA (2020h). Talks. `http://www.oia.unsam.edu.ar/charlas/`. Accessed: 2020-02-08.

OEI (2020). Organization of iberoam erican states – ciic. `https://www.oei.es/historico/ciic.htm`. Accessed: 2020-01-18.

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, 207–222.

Kalinicenko, E. (2020). *International Olympiad in Informatics – statistics*. `http://stats.ioinformatics.org/results/ARG`. Accessed: 2020-01-18.

Laaksonen, A. *Competitive programmer's handbook*. 2017.

Maggiolo, S., Mascellani, G. (2012). Introducing cms: A contest management system. *Olympiads in Informatics*, 6.

Maggiolo, S., Mascellani, G., Wehrstedt, L. (2014). Cms: a growing grading system. *Olympiads in Informatics*, 8.

---

* OIA – Argentine Computer Olympiad
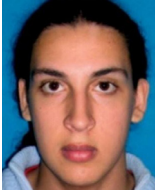
A.S. Gutiérrez. MS in Computer Science at FCEN-UBA (Buenos Aires University). Two times IOI contestant (2006 and 2007), winning a Silver Medal in 2007. Two times World Finalist and Latin American Champion at ICPC (2009 and 2011), and seven times world finalist as a coach (2010, 2013, 2014, 2015, 2016, 2017 and 2020), coaching the Latin American Champion team in 2015. Actively involved in helping with the organization and running of the Argentine Olympiad in Informatics (OIA) contests since 2011. Since 2015, official National Jury and IOI team trainer at OIA, and Deputy Leader of Argentina at IOI. Organizer in 2019 and 2020 of the Iberoamerican Olympiad in Informatics (CIIC), a regional contest spanning Latin America, Spain and Portugal. More than ten times organizer and lecturer at more than five different ICPC Training Camps in Argentina and other Latin American countries. Teaching fellow from 2013 to 2018 at FCEN-UBA, teaching the course "Algorithms and Data Structures 3" (Algorithmic techniques and graphs). More than five year experience as a software engineer developing customized search engines and algorithms for an online retail business at Brightsector Algorithms. Assisted Topcoder as local organizer of the TCO 2018 Argentina Regional Event. Native Spanish speaker, fluent in English, studying Russian (approaching A2 level).

Competitive Programming 4: The New Lower Bound of Programming Contests in the 2020s

Steven HALIM

School of Computing, National University of Singapore
Computing 1, 13 Computing Drive, Singapore 117417
e-mail: dcssh@nus.edu.sg

Abstract. Seven years have passed since me and my brother Felix Halim released the 3rd edition of our Competitive Programming book (CP3) on 24 May 2013 that had influenced the competitive programming field in the past decade: 2010s. We have just released the 4th edition of our book (CP4) on 19 July 2020 – the original IOI 2020 arrival day where free preview copies should have been given to all invited delegations. In this short report, we address two groups of readers: those who have read/heard about CP3 and those who are new with this book.

Keywords: competitive programming, book, IOI, ICPC.

1. The Impact of the Earlier Editions of Competitive Programming Book

We first released Competitive Programming 1 (CP1) before IOI 2010 in Waterloo, Canada, and updated it one year later with the 2nd edition (CP2) after IOI 2011 in Pattaya, Thailand. However, the most impactful and long-lasting edition so far was the 3rd edition (CP3), released in 2013 before IOI 2013 in Brisbane, Australia (Halim and Halim, 2013). Before we wrote CP1, there is only one other existing English book in competitive programming: *Programming Challenges* (Skiena and Revilla, 2003).

We write Competitive Programming book with the main objective of improving the lower bound of the typical long tail of the distribution of worldwide (pre-)competitive programmers, i.e., secondary/high school students or freshmen in University who have just started basic programming and want to improve their data structures, algorithms, (competitive) programming techniques, and problem-solving skills.

In these past seven years (2013–2020), we have received numerous thank you emails (see selected testimonials at `https://cpbook.net/`) and also the annual requests for autographs and/or wefie whenever we met young CP3 readers in the recent IOIs or ICPC Regionals/World Finals. Many of these young readers found CP3 as a "catalyst" for

their personal competitive programming growth. By reading CP3, these young readers can quickly learn about the knowledge needed to compete decently in the IOIs, e.g., the IOI syllabus (Forišek, 2019) and in the ICPC regionals. While not the original intention, many readers have also expressed gratitude that studying the material in CP3 helped them to secure lucrative jobs at top IT companies. The generally positive feedback from the readers motivated us to continue studying the recent trends in this fast-evolving competitive programming world, including to read the *Guide to Competitive Programming* book (Antii Laksonen, 2017 and 2020), and to continue this book writing project with a third author: Suhendry Effendy. Now in the year 2020, we are ready to release our latest/4th edition (CP4).

In the next section, we highlight the main differences between this impactful CP3 with the new CP4 for the new decade: 2020s.

2. Comparison with CP3 (2013)

This section is for the benefit of the reader who has read CP3 sometime in the past seven years and possibly a current active IOI/ICPC coach, a high school teacher in informatics, or a University lecturer that can influence future young CP4 readers.

The major change is the split of CP4 into two volumes: Book 1 is mostly for IOI (most content of the IOI syllabus (Forišek, 2019) are discussed here) + basic ICPC level and Book 2 is mostly for medium ICPC level. Secondary or high school students can start from CP4 Book 1 first and only move to Book 2 when they enter University.

Other major update is the addition of Kattis online judge (`https://open.kattis.com`). Kattis has growing problem bank that includes IOI-related tasks, e.g., Croatian Open Competition in Informatics (COCI) tasks 2006–2017, including some newer problem types: interactive and constructive problems.

Features	CP3	CP4
Number of Books	1	Split into two books
Number of Chapters	9	1–4 + 5–9 = 9
Number of Pages	447	329 + 352 = 681 (> 1.5x)
Authors	Steven, Felix	Steven, Felix, Suhendry (+1)
Authors combined experiences	ex IOI/ICPC Regionals+ World Finals Contestants, active coaches and problem authors of recent programming contests	+ ICPC Asia Singapore Regional Contest Director (2015+2018), ten ICPC Asia Regional wins (as coach), many more IOI gold+ silver+bronze medals for team Singapore, IOI Deputy Director+ International Committee member (2020+2021)
Programming Languages	C++ (11), Java (7)	C++ (17), Java (11), and +2 more: Python (3), OCaml
Programming Exercises	UVa: 1675	UVa+Kattis (n: 3458 (> 2x)
Figures generated using	PowerPoint + older visualization tool	Mostly VisuAlgo (Halim, 2015) screenshots

Albeit not included in the IOI yet (but available in the ICPC), we have integrated discussion on when it is appropriate to use Python (3) programming language to solve competitive programming problems whenever it is allowed. Sample implementation code discussed in CP4 are available at `https://github.com/stevenhalim/cpbook-code`

We also rewrite most sections and update many figures with the latest screenshots of our Visualization tool: VisuAlgo: `https://visualgo.net` (Halim, 2015), thus CP4 readers can further deepen their understanding of the data structure/algorithm being discussed by trying their input data at VisuAlgo together when reading CP4.

The major additions are the topics that were previously not written yet in CP3 but are now written in CP4 (many are outside the IOI syllabus (Forišek, 2019) and more appropriate for ICPC): Modular Multiplicative Inverse, String Hashing, Square Root Decomposition, Heavy-Light Decomposition, Tree Isomorphism, De Bruijn Sequence, Fast Fourier Transform, Chinese Remainder Theorem, Lucas' Theorem, Combinatorial Game Theory, Egg Dropping Puzzle, Dynamic Programming Optimization, Push-Relabel algorithm, Kuhn-Munkres algorithm, Edmonds' Matching algorithm, Constructive Problem, Interactive Problem, Linear Programming, Gradient Descent.

3. For New CP4 Readers in 2020s

The 2020s decade has just started (albeit with a global COVID-19 pandemic). We believe that CP4 Book 1 provides the necessary (but not necessarily sufficient) conditions needed to prepare the many young readers for their National Olympiad in Informatics (NOI) preparation leading to the IOI. Similarly, we also believe that CP4 Book 1+2 provides the same necessary (but not necessarily sufficient) conditions needed to prepare many new University students for their ICPC Regionals leading to the ICPC World Finals.

The details on how to get a copy of this book via its various distribution channels can be found at the book's companion website: `https://cpbook.net`.

References

Forišek, M. (2019), `http://people.ksp.sk/~misof/ioi-syllabus/`

Halim, S., Halim, F. (2013). *Competitive Programming 3: The New Lower Bound of Programming Contests*. Lulu

Halim, S. (2015). VisuAlgo – Visualising Data Structures and Algorithms Through Animation. *Olympiads in Informatics*, 9, 243–245

Laaksonen, A. (2017). A new book on competitive programming. *Olympiads in Informatics*, 11, 167–170

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

Skiena, S.S., Revilla, M.A. (2003). *Programming Challenges : The Programming Contest Training Manual*. Springer

S. Halim is a senior lecturer in the School of Computing, National University of Singapore (SoC, NUS). He teaches several programming courses in NUS, ranging from basic programming methodology, intermediate to hard data structures and algorithms, web programming, and Competitive Programming. He is the coach of both the NUS ICPC teams and the Singapore IOI team. So far (2009–2019), he and other trainers at NUS have successfully groomed various ICPC teams that won ten different ICPC Regionals, advanced to ICPC World Finals eleven times with the current best result of Joint-14th in ICPC World Finals Phuket 2016, as well as seven gold, nineteen silver, and fifteen bronze IOI medalists. He is also the Regional Contest Director of ICPC Asia Singapore 2015+2018 and is the Deputy Director+International Committee member for the IOI 2020+2021 in Singapore.

IOI Talks:
New Initiative for Publishing Presentations, Events, Interviews, Book Recommendations and Videos of Interest to the IOI Community

Bojan KOSTADINOV[1], Mile Jovanov[2]

[1]*University American College Skopje*
 III Makedonska Brigada, 60, 1000 Skopje, Macedonia
[2]*Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University*
 st. Rugjer Boshkovikj 16 Skopje, Macedonia
e-mail: bojan.kostadinov@gmail.com, mile.jovanov@gmail.com

Abstract. Most of the activity of the IOI community occurs during the International Olympiad in Informatics (and a couple of weeks around the event). In this report we present "IOI Talks" which is a new initiative to occasionally publish presentations, events, book recommendations, interviews and videos that can be of interest to the broader IOI community. Additionally, we will give an overview of the content for the first event.

Keywords: IOI, IOI talks, competitive programming, IOI event.

1. The Motivation for "IOI Talks"

For the most part, the activity of the IOI community occurs during the International Olympiad in Informatics. This was the main parameter that we had in mind when we proposed a project on the IOI Call for Projects 2019/2020, to organize an online event (codename "IOI Weekend", which was later updated to "IOI Talks"). The key idea was to organize one or more events for the IOI community that will feature presentations, interviews, tasks, videos, book and site recommendations and more. This event (or events), timewise, will not be tied to the IOI and will be organized during a completely different period of the year – i.e. the idea is to keep the community active outside the actual IOI competition.

The Initial Format of "IOI Talks"

Originally, the plan was to stream the events on YouTube, and feature video interviews (and presentations) with past successful IOI participants, talks with other people that will be interesting to the IOI community (like, for example, people who can present how to move from competitive programming into a successful software development job), presentations by book authors, talks on what IOI Hall of Famers are doing at the moment (or how they are involved with the IOI), videos with highlights from previous IOI Olympiads, notes about the next Olympiad (locations, photos, how the organization is going, etc), then presentations on how to get started with competitive programming and the International Olympiad in Informatics (i.e. where to learn programming, with some examples of easier IOI subtasks), and more. The plan also featured Q&A sessions and gathering feedback from participants via an online form.

As part of the application for the IOI Call for Projects 2019/2020, it was indicated that the resulting content will be of very high quality, and highly interesting and engaging for the community involved with the International Olympiad in Informatics. The plan outlined that we will be targeting and preparing videos that are interesting to both competitors, teachers, and team leaders. Everything outlined above, including contacting content editors, past contestants, and interviewees, as well as video production and hosting, was to be delivered as part of this project.

Clearly, it is possible to organize even more presentations (for example, gathering questions, presenting discussions and answers to those questions by IC members or the IOI president), as well as other content, topics or features depending on interest and availability.

Finally, as mentioned in the previous sections, one of the main goals of the project is to engage the competitive programming community (and especially people interested in the IOI) outside the Olympiad itself – so that there are discussions, surveys, talks and buzz around the International Olympiad in Informatics outside the month when the actual competition takes place.

The Final Format of "IOI Talks" and its 2020 Edition

After the acceptance of this project, as well as the positive feedback and response from the IC committee and other parties who were contacted to gather their early thoughts, we decided on a slightly different format than initially proposed.

Currently, the content of the first IOI talks is reachable by registering on the IOI website by following this link: `https://ioinformatics.org/event/july-2020/signup`, and some of the interviews are directly available on the homepage of the International Olympiad in Informatics.

The final implementation aims to give team leaders and other members of the IOI community as much freedom as possible, so after registering, they get a link that allows them to view the content themselves, share the link with their team or friends so they can view the talks and presentations on their own time, or (what we recommended in the announcement of the project) is to use the content and videos during other contest/ camp activities and events to stimulate interest in the IOI, or to create an online event where people watch or go through the content together (we didn't want to stream everything at a specific time/day, because of all the different countries and time zones – and instead wanted to give everyone an opportunity to organize things themselves, and to be able to watch any of the videos/interviews in any order they want). We might reconsider a different approach/strategy in the future.

In the first edition of IOI Talks, we featured interviews and presentations with Benjamin Qi – the winner of IOI 2018 and 2019, Martin Mares – the chair of the ITC, Petr Mitrichev – one of the best competitive programmers, IC members, various discussions on topics such as introducing Honorable mentions at the IOI, a booklet featuring solutions to past IOI problems (IOI 2018), and more. We would like to thank everyone who participated.

In the registration form and on the event page, participants are allowed to share any feedback or comments they might have. We would highly appreciate input from the readers. The initial feedback from the other participants is very encouraging.

Finally, we can proudly announce that although the IC of IOI approved funding for this project, we have implemented the first edition of IOI Talks for free, using our personal funding. Similarly, the work that was done as part of this project will help the actual website of the International Olympiad in Informatics in the near future, as it now supports publishing interviews, videos, streaming, event content with feedback, and a lot more. If you like to be featured in an interview, prepare a video presentation or share anything else you believe would be interesting to the IOI community, feel free to send us an email.

References

IOI Call for Projects 2019/2020. (2019),
   https://ioinformatics.org/news/call-for-projects/17
Official website of the IOI,
   https://ioinformatics.org
Signup form for IOI Talks #1 (2020).
   https://ioinformatics.org/event/july-2020/signup
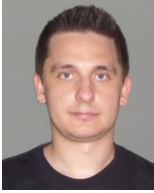
B. Kostadinov is the founder of Cloud Solutions, an author, and a former competitive programmer. In 2014, he defended his MSc thesis in Intelligent information systems at the Faculty of Computer Science and Engineering, University "Ss. Cyril and Methodius", in Skopje. He is one of the organizers of the national competitions in informatics in Macedonia, and the Beaver event.

M. Jovanov is an associate 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, and an IC member at IOI since 2015. His research interests include development of new algorithms, future web, and informatics education.

National Programming Competitions, Team Selection and Training in Hungary

László NIKHÁZY[1], László ZSAKÓ[2]

[1]*Doctoral School, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary*
[2]*Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary*
e-mail: nikhazy@inf.elte.hu, zsako@caesar.elte.hu

Abstract. Programming competitions for middle and high school students have a long tradition in Hungary, the first national competition dates back to 1985, and our country has been participating in the International Olympiad in Informatics (IOI) from the very beginning. This paper presents the current situation and challenges of Hungarian nationwide programming contests, national team selection, and training system.

Keywords: programming competition, informatics education, training, IOI, Hungary.

1. Introduction

In Hungary, already in the early eighties, regional programming competitions were organized for middle and high school students. As a result of the school computer program and the evolving microcomputer clubs, the Nemes Tihamér Competition was established in 1985, being the first national programming competition. In the first year, approximately 1000 students entered the competition. The number of participants then gradually increased, in recent years there have been around 2500–3000 students taking part, from 200–250 schools regularly.

Since then, numerous other informatics competitions have come into existence. Not only in competitive programming, but also in robotics, application development, application usage, and computational thinking. We provide an overview of them in the next chapter of this article. We describe the details of the programming category of the Nemes Tihamér Informatics Competition (NT) and the National High School Competition in Informatics (OKTV) in the third chapter.

Hungary has participated in the IOI every year since its start and won 13 gold, 31 silver and 44 bronze medals in total. Our country hosted the competition already once in 1996 and we are going to host it for the second time in 2023. It will be a special year for the Hungarian computer science community, we celebrate the 120th birth anniver-

sary of John von Neumann, the greatest Hungarian figure in the theory of computing, who is often regarded as the foremost mathematician of his time.

Hungary was one of the initiators of the Central European Olympiad in Informatics (CEOI), and our team participated every year since the start of the competition in 1994. The CEOI was held in Hungary four times already (1995, 2001, 2005, 2012), and we are hosting the competition this year, in 2020, in the city of Nagykanizsa. Hungarian contestants have also achieved excellent results at the CEOI, they have won 2 gold, 14 silver and 42 bronze medals in total. Our country took part in all three editions of the European Junior Olympiad in Informatics (EJOI) (Manev & Yovcheva, 2017) so far, and we are also going to participate in the European Girls' Olympiad in Informatics (EGOI) starting from its kick-off in 2021.

The IOI and CEOI team selection process have long traditions, the students qualify through an extensive competition, which we describe in the fourth chapter. In recent years, a similar, but shorter competition has been organized for the EJOI team selection. In the fourth chapter, we also provide an overview of the training and team preparation system currently in Hungary. Finally, in the fifth chapter, we write about the challenges we are facing at present and our plans for the future.

2. National Competitions in Informatics

There are several national competitions in informatics for primary, middle, and high school students, not limited to computer programming. We categorize them in the table below. Their websites are listed at the end of the article.

We would like to mention that in some cases the categorization is not straightforward. For example, a couple of application development competitions have some algorithmic challenges and similar input-output testing as the IOI-style competitions, which focus on algorithms and data structures. Furthermore, many of the first rounds of IOI-style competitions test algorithmic thinking with tasks on paper, so we put the first rounds separately to the corresponding category. In the next chapter, we focus on IOI style national competitions, which are the first steps towards the national team selection. We provide a very short description of the other competitions here.

The Nemes Tihamér Online Programming Competition is a five-round online contest which is a practice and preparation opportunity for the onsite competition.

The Programming Contest of the University of Debrecen is a team competition for high school students similar to the International Collegiate Programming Contest (ACM ICPC).

The Izsák Imre Gyula Complex Science Competition is a multi-discipline contest in mathematics, physics, and informatics. It involves a programming contest and the winner of that category is invited to the IOI Qualification Competition.

The Logo National Informatics Competition is organized in five age groups for elementary, middle, and high school students. The main topic is turtle graphics, except for category 0, which is a robot programming contest for elementary school pupils.

Table 1

National informatics competitions in Hungary

IOI-style	Nemes Tihamér Informatics Competition (programming category)
	National High School Competition in Informatics (programming category)
	Nemes Tihamér Online Programming Competition
	Programming Contest of the University of Debrecen
	Programming category of the Izsák Imre Gyula Complex Science Competition
Algorithmic thinking	Logo National Informatics Competition
	First round of the Nemes Tihamér Informatics Competition
	First round of the National High School Competition in Informatics
Computational thinking	e-Hód Competition (Bebras)
Robotics	First® LEGO® League
	World Robot Olympiad™
	Category 0 of the Logo National Informatics Competition
	Robot Sumo Competition
	Robot Programming National Team Contest
Application development	Neumann János Talent Search Software Product Competition
	Programmers' National Dusza Árpád Memorial Competition
	ProgRace Programming Contest
	PENDroid High School Competition
	B³ – Bakonyi Bitfaragó Championship
Application usage	Nemes Tihamér Informatics Competition (application category)
	National High School Competition in Informatics (application category)
	Kozma László National Informatics Competition

The Bebras challenge is called e-Hód Competition in Hungary. It is gaining huge popularity, in 2019 there were more than 27 thousand participants from 202 schools across all age groups.

There are regional and national rounds in Hungary of the well-known international robot programming competitions, the First® LEGO® League (FLL), and the World Robot Olympiad™ (WRO). The FLL is held since 2004, while the WRO was first organized in Hungary in 2014, and our country hosted the WRO 2019 Final in the city of Győr.

In the Robot Sumo Competition, two-person teams have to build LEGO® robots that fight against each other like the sumo sport. In the Robot Programming National Team Contest, teams of two or three middle school students compete in designing and programming a LEGO® robot to solve challenges on a field.

In the Neumann János Talent Search Software Product Competition there is no task, but contestants can present their work to the jury, and get prizes in multiple categories: applications, games, hardware control, computer graphics, animations, and computer-aided design.

In the Programmers' National Dusza Árpád Memorial Competition every team has to write an application solving the same problem. It has a regular PC software development, web development, and mobile application development category as well.

The ProgRace Programming Contest is organized by the University of Pécs for both high school and university students. Teams of two or three contestants can enter using any programming language. They have to solve an open-ended challenge from different topics each year.

The PENDroid High School Competition is an Android app development competition organized by the Nagykanizsa Campus of the University of Pannonia. There is a qualification round that has educational purposes, and a final round, where the teams should bring an Android game written by them and they also have to solve a task onsite.

The B³ – Bakonyi Bitfaragó Championship is hosted by the Faculty of Information Technology of the University of Pannonia. Teams of high school students compete in several categories, and the tasks usually involve web programming.

The Nemes Tihamér Informatics Competition and the National High School Competition in Informatics have an application usage category, where contestants need to solve tasks in office applications. Tasks involve documents, spreadsheets, presentations, static web pages, database management, and image editing.

The Kozma László National Informatics Competition is a team contest for middle and high school students, where participants create solutions to project-based tasks in office applications.

3. The Nemes Tihamér Programming Competition

The NT competition and the OKTV are the most prestigious programming contests in Hungary, which also serve as the primary entrance to the qualification competitions of all the international Olympiads. NT is organized by the cooperation of the John von Neumann Computer Society (NJSzT) and the Eötvös Loránd University (ELTE), and the OKTV is organized by the Education Office of Hungary together with ELTE. They have the same format, same submission system, and most **members of the scientific committee** work on both competitions.

As mentioned above, the NT competition was first organized in 1985. It had two rounds for 9 years, the first round was written on paper in schools, and the second round held at ELTE, where competitors usually had to write a program for a single major task, in BASIC, Pascal, or possibly C of their choice. From 1990, the 9–10th grade (15–16 years old) students are given a different set of tasks than the 11–12th graders (17–18 years old). From 1994, the competition was expanded with another category, students from the 5–8th grade (11–14 years old). In 2003, the third age group (11–12th grade) of the NT competition has got the official status of the National High School Competition in Informatics (OKTV). A few years ago, a parallel category was introduced for the 11–12th grade students, the age group 3 of the NT competition, where students can compete who are not eligible to enter the OKTV, together with the contestants who do not meet the very strict limit for advancing in the OKTV.
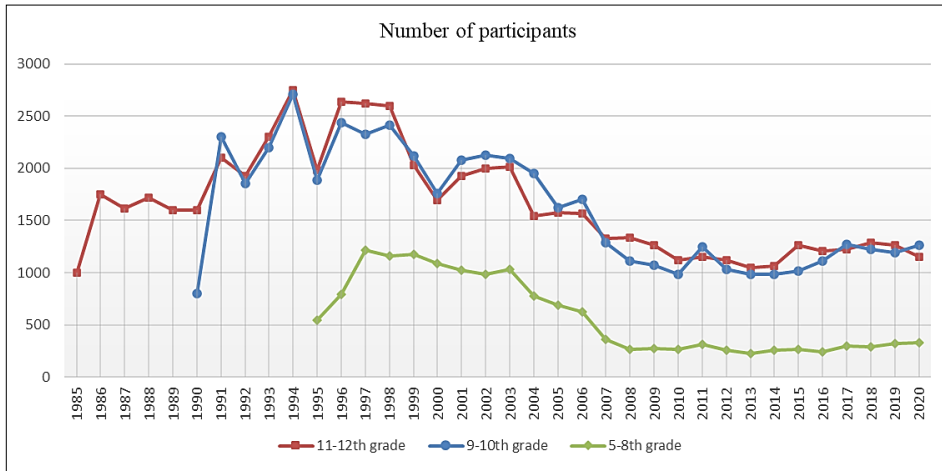
Fig. 1. Number of participants yearly in the Nemes Tihamér Competition.

An online judge system has been used at the competition in recent years, which was developed for CEOI 2012. Before that, contestants had to hand in their solution programs at the end of the contest (through the local network, or floppy disks), and they were evaluated afterward with testing scripts.

At present, there are three stages of the competition, a school round, a regional round, and a final. All rounds are held at the same time for all contestants, with the same, centrally composed task sheet. In the first round, everyone writes the contest at their own school, the second round is organized by selected schools in each region. The final takes place in 3–5 cities in Hungary since the online judging makes it easy to organize the contest in different locations. A few years ago, everyone had to travel to the capital. It is still the case for the OKTV category, as it is regulated by the ministry.

3.1. *School Round*

The first round usually takes place in early November. No computers are used at this stage, students have to solve 4–8 tasks on paper. (More precisely, the children in the lowest age group can choose to solve one task on the computer, but the rest of them are on paper.) The tasks are designed to test algorithmic thinking and understanding topics in computer science, particularly, but not limited to algorithms, data structures, and programming. The algorithms are described with a Hungarian pseudocode syntax close to natural language. There can be problems from other areas of computer science, like formal grammars, automatons, logical circuits, functional and logical programming languages, etc. In this case, the task description contains enough information about the topic to solve the problem, since these areas are not taught in the school. These problems test understanding new concepts, besides analytical thinking.

We find it beneficial to have such a round, because (1) it promotes computational thinking, (2) students can enter without knowing a programming language, and (3) it provides the same circumstances for everyone, no matter how equipped their school is. We can reach out to a lot more students allowing them to enter without experience in computer programming, and thus their talent in this field can be discovered.

3.2. *Regional Round*

The second round is typically held in January with around 600 competitors altogether in the three age groups. Students have to solve IOI-style tasks on the computer. Tasks are assembled by the jury in a way that the best contestants can be selected based on the results, while everyone should have some sense of achievement. In the second and third age group (grades 9–12) there are 5–7 tasks for 5 hours, while in the first age group (grades 5–8) there are 3–4 tasks for 3 hours.

Contestants can submit their code multiple times in the online judging system, and it gets scored instantly using input-output testing. The submission with the maximum score is considered. The score for a task is calculated as the sum of the scores for the test cases passed, similarly as the IOIs before 2005 (Verhoeff, 2009). Each test case has an assigned score that can vary. If there are multiple expected outputs (e.g. the optimum value and the way to achieve it) the solution to them can be scored separately. We create subtasks by indicating the number of points that can be achieved with smaller limits or more specific constraints than the general ones in the task description. The time limit varies depending on the task, the most typical values are between 0.2–0.6 seconds.

There is a wide variety of programming languages allowed: C/C++, Java, C#, Python, Pascal, and Visual Basic. It remains a challenge for the task writers to achieve that the expected solutions meet the same time limit in all languages, while less optimal solutions should not pass in any language. But even like this, every language has its advantages or disadvantages for certain tasks. Contestants need to be aware of this issue when choosing the programming language. In fact, they can choose different languages for each task.

3.3. *Final Round*

The final regularly takes place in March. There are 40–60 participants in the first age group, 80–100 in the second age group, and 50 in the OKTV category plus 30–50 in the third age group. The final has the same format as the second round, but the problems are more difficult, more advanced topics are included. The tasks are designed to make a difference between the very best students as well. The contest duration is 6 hours for the second and third age group, and 3 hours for the first age group. The number of tasks is about the same as the second round.

The first 10–15 students of each age group of the NT competition are awarded a book. Contestants in the first 3 places of the OKTV get awards, and a so-called Tal-

ent Passport that provides entrance to talent education programs during their university studies, and the first 30 students get extra points for their application to universities. The best contestants in each category get invitations to the Qualification Competition for International Olympiads.

3.4. *Topics of the Competition*

The topics of the problems in the regional and final round are a subset of the IOI Syllabus (Verhoeff, Horváth, Diks, & Cormack, 2006). We are in the process of creating a similar document called NT Syllabus (Nikházy, 2019) to specify exactly what contestants need to know and thus help them and their teachers in preparing for the contest. Another goal of the syllabus is to provide guidelines for task setters in creating appropriate tasks. It contains the required knowledge for each age group of the national competition. For the IOI team selection competition, the IOI Syllabus fills this role. To create the syllabus, we examined problems of the last 5 years and created statistics about the topics appearing in problems of the regional and final round. In the following, we provide a brief overview of the topics by age group.

The problems for 5–8th grade students essentially do not include complex "textbook" algorithms to be mastered in advance. About 65% of the tasks can be solved using the so-called patterns of algorithms (Szlávi & Zsakó, 2008). These are simple algorithm templates that students can use in many cases without their awareness, for example, maximum selection, linear search, counting, filtering. Roughly 35% of the tasks require the use of some more advanced algorithmic strategy, for example, greedy algorithms, dynamic programming, and the two pointers technique.

In addition to the above-mentioned topics, graph algorithms appear in tasks for 9–10th grade students and this is a major change compared to the first age group. According to the statistics, about 30% of the problems include graphs, which means that there is usually at least one such task in the regional round and two in the final. There is much more emphasis on dynamic programming (DP) than for the younger students. Problems in enumerative combinatorics frequently appear in the competition because students acquire the necessary mathematics knowledge at this age. Simulation is an important technique, the difficulty of such tasks lies in the complex implementation.

There are some significant enhancements in the topics for the 11–12th grade students compared to younger ones. More advanced graph algorithms are required, like strongly connected components, articulation points and bridges, Eulerian path, shortest paths, and minimum spanning tree in weighted graphs. (Weighted graphs are included for the second age group, too, but they appear less frequently.) There is a completely new domain, geometric algorithms, which often provide the most difficult problems. Utilizing advanced data structures like the priority queue (heap), the set and map from the standard library are required to perform successfully in this age group. In conclusion, we can say that the topics of the OKTV are a subset of the IOI Syllabus with only some of the hardest algorithms and data structures excluded.
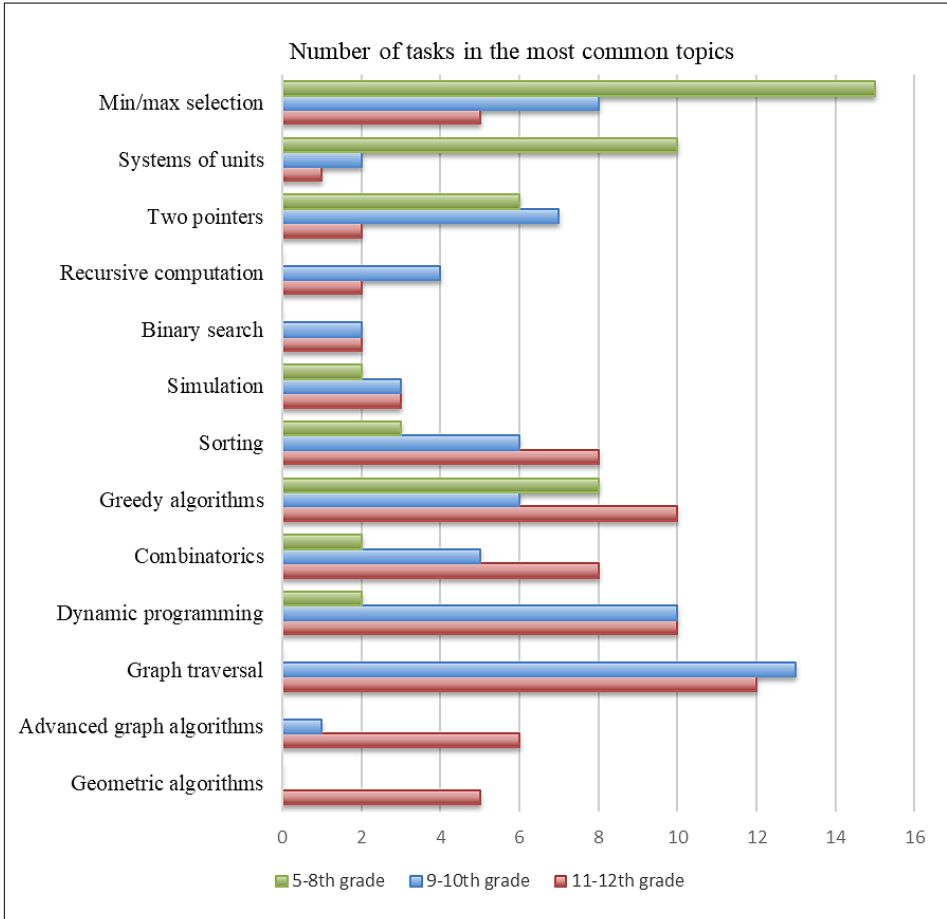
Fig. 2. Appearance of the most significant topics in the last 50 tasks of each age group.

4. Team Selection and Training for the IOI, CEOI, and EJOI

4.1. *Qualification Competition*

There is an intensive and comprehensive Qualification Competition (QC) to select the Hungarian team for the IOI and CEOI. There are 6 rounds, each round has 3 problems for 3 hours, which means there are 18 problems in total. Usually, two rounds are held on one day, so there are 3 competition days. The topics of the competition are the same as at the IOI, the IOI Syllabus is used as the definitive guide to propose tasks. In each round, there is one problem that tests knowledge of some standard algorithm or a typical problem-solving strategy, one intermediate, and one hard task which are similar to CEOI

and IOI tasks. For evaluation, the same system is used as the national competition, but we try to put emphasis on subtasks by creating more subtasks for one problem and also grouping more test cases into one test file when we find it necessary.

The 4 students of the IOI team and the 4 students of the CEOI team are selected based on their total points in the 6 rounds of the QC. Students in the 10–11th grade can be members of both the IOI and CEOI team, younger students are only included in the CEOI team, and 12th-grade students only in the IOI team. This is to provide more opportunities for younger students for participating in international contests.

Students are invited to enter the contest based on their results in the national competition (OKTV and NT). In total, the competition is started with approximately 25–30 students in both the IOI and CEOI categories, about 70% of the participants are common. After the first two rounds, around 15 contestants advance in each category and after the second two rounds, only about 10 students per category compete in the last two rounds for the 4–4 places.

For the EJOI, we have a separate, shorter qualification contest, because we need to test different knowledge of the younger children. There are 2 rounds in this contest, each round has 3 problems for 3 hours. About 12–16 students are invited to the first round, and 6–12 participants advance to the second round, after which the 4 students of the EJOI team are selected based on their total score.

The Fig. 3 shows an overview of the system of competitions presented in this article. We included the trainings that help students in preparing for these contests. The next chapter provides more details about them.
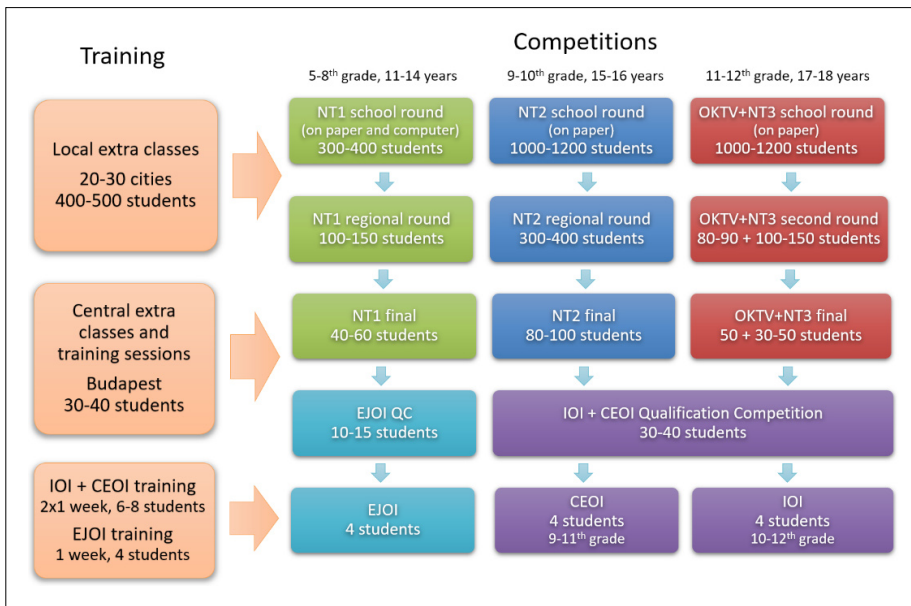
Fig. 3. Training and competitions in Hungary on the way to International Olympiads.

4.2. *Training*

After the IOI and CEOI teams are selected, there is twice a one-week long training for team members. These are usually held in May and June, at ELTE, Budapest. We cover advanced topics of the IOI Syllabus and interesting tasks from past IOI and CEOI competitions. The training material is changing each year, customized to the skills and knowledge of the team members. The work is very intensive, 8 hours a day with a lunch break. Other students, who are not team members but have a good chance to qualify next year, are also welcomed to join the training.

The EJOI team has a separate one-week training, usually organized in July. The format is the same as above, but the topics are naturally different: graph algorithms, dynamic programming, greedy algorithms, recursion, backtrack, basic combinatorics. We also teach best practices and little tricks of every subject matter.

Hungarian students have been very successful in international competitions considering that we are a small country of 10 million people. The Fig. 4 shows the medals won at the IOI: a total of 13 gold, 31 silver and 44 bronze medals (data from IOI Statistics website).

Above we described the training specifically for the teams going to international Olympiads. There is also a countrywide system of talent education in Hungary, coordinated by the Talent Education Department of NJSzT and ELTE. The first stage is organized locally by volunteering schools in 20–30 cities across the country. They give extra courses to middle and high school children, who are interested in computer programming. The topics are flexible, teachers can conduct these classes according to their knowledge and interests. They are not limited to competitive programming, for example, there are some classes about robotics, application development, and a lot of introductory programming courses with block-based coding. ELTE and NJSzT provide education materials. The topics of the education materials include basic algorithmic patterns (sum, count, search, filter, maximum selection, sorting, merging, intersection, union, recursion) and introductory algorithmic strategies (greedy algorithms, backtracking, divide and conquer).
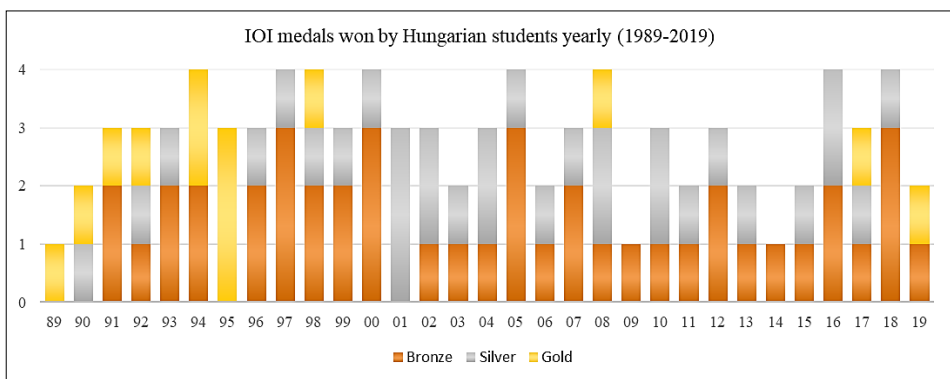
Fig. 4. Medal distribution of Hungarian students at the IOI yearly (1989–2019).

As a continuation, there are 5 training sessions yearly for the most talented students of the country, held in Budapest by one of the authors, László Zsakó. Students can apply based on their teachers' recommendations. In these training sessions, the following topics repeat every two years: greedy algorithms, recursion, dynamic programming, trees and binary trees, graph algorithms, combinatoric algorithms, geometric algorithms, and simulation.

Additionally, there are some extra courses held by university students in selected schools in Budapest. The first author of the article is currently involved in one of them. The classes are bi-weekly and open for all talented students from the capital and other cities, too. We solve former IOI tasks and learn advanced algorithms and data structures, according to the level of the participants.

There are few cities apart from Budapest where some prominent teacher leads a self-developed talent education system. The results of the programming competitions show the existence of such places. For example, the city of Nagykanizsa is providing numerous good contestants every year, due to the excellent work of Ágnes Erdősné Németh, who thereby became vice president of NJSzT recently. Other notable places of good programming education include Debrecen, Zalaegerszeg, Szatmárnémeti, and there are several individual teachers scattered in the country.

5. Current Challenges and Plans

Although there is a talent education network established in Hungary, on the highest level, the achievements of students rely on a local excellent teacher in very few places, or completely on their individual efforts. We would like to start a program focused on the most talented students countrywide, forming groups that stay together for years, organizing weekend camps, and online workshops for them. The bases of this program were outlined in (Nikházy, 2020). A similar system (Győri & Juhász, 2017) was built in mathematics by The Joy of Thinking Foundation that we consider as a model. We want to put emphasis on creating a community. Currently, gifted students mostly meet only at the competitions. In our experience with the mathematics camps, personal connections play a big part in the engagement of learning at this very high level.

We also plan to start talent training programs focused on girls. The underrepresentation of women in IT is a well-studied topic worldwide. Unfortunately, Hungarian programming contests also show this phenomenon. With Hungary participating in the EGOI from its start in 2021, we are committed to encouraging girls to master programming. While we include girls in the above-mentioned talent programs, we would like to organize training specifically for girls, where more than half of participants are young ladies, and the rest of them are boys with similar knowledge level.

There is room for improvement in the competitions as well. The top contestants' knowledge level is getting higher year by year, and we, the scientific committee of the NT competition, try to keep up with it while maintaining the spirit and fairness of the contest. Since there are a wide variety of languages allowed, it is a challenge to create language-independent tasks, where the evaluation makes it possible to differentiate

between the solutions according to their computational complexity with the same time limit for every language. It is not only our problem, but many other systems also face this issue. We are looking into solutions, we might consider language-dependent time limits. The current scoring system, which is based on individual test cases, also makes it hard to rate solutions according to their complexity, because sometimes only specially constructed tests detect the slowness of an algorithm. With a lot of other tests in place, a suboptimal solution can get almost full points. Scoring based on test groups is an option that we are looking into currently, but we also find it unfair to give no score to a solution that is correct for most of the test cases (Erdősné Németh & Zsakó, 2018).

Finally, it is always our goal is to make programming competitions more popular in Hungary. We hope that our current and future work in talent education and community building will encourage more and more students to learn programming and participate in contests.

References

Erdősné Németh, Á., & Zsakó, L. (2018). Grading Systems for Algorithmic Contests. *Olympiads in Informatics, 12*, 159–166. DOI:10.15388/ioi.2018.13

Győri, J., & Juhász, P. (2017). An extra-curricular gifted support programme in Hungary for exceptional students in mathematics. In *Teaching Gifted Learners in STEM Subjects* (pp. 89–106). Routledge.

Manev, K., & Yovcheva, B. (2017). First European Junior Olympiad in Informatics. *Olympiads in Informatics, 11*, 171–173. DOI:10.15388/ioi.2017.15

Nikházy, L. (2019). A Nemes Tihamér Programozási verseny témaköreiről készült syllabus. *InfoDidact2019*. Zamárdi, Hungary: Webdidaktika Alapítvány.

Nikházy, L. (2020). A Problem-based Curriculum for Algorithmic Programming. *Central-European Journal of New Technologies in Research, Education and Practice, 2*(1), 76–96. DOI:10.36427/CEJNTREP.2.1.399

Szlávi, P., & Zsakó, L. (2008). *Módszeres programozás: programozási tételek.* Budapest: ELTE Informatikai Kar.

Verhoeff, T. (2009). 20 Years of IOI Competition Tasks. *Olympiads in Informatics, 3*, 149–166.

Verhoeff, T., Horváth, G., Diks, K., & Cormack, G. (2006). A proposal for an IOI Syllabus. *Teaching Mathematics and Computer Science, 4*(1), 193–216.

Websites

B³ – Bakonyi Bitfaragó Championship: `https://verseny.mik.uni-pannon.hu/`
EGOI: `https://egoi.org/`
EJOI: `http://ejoi.org/`
e-Hód Competition: `http://e-hod.elte.hu/`
First® LEGO® League: `https://www.firstlegoleague.org/`
IOI Statistics: `https://stats.ioinformatics.org/`
IOI Syllabus: `https://ioinformatics.org/page/syllabus/12`
Izsák Imre Gyula Complex Science Competition: `http://www.zmgzeg.sulinet.hu/izsak/`
Kozma László National Informatics Competition: `https://isze.hu/kozma-laszlo-orszagos-informatika-alkalmazoi-tanulmanyi-verseny/`
LOGO National Informatics Competition: `http://logo.inf.elte.hu/`
Nemes Tihamér Informatics Competition: `http://nemes.inf.elte.hu/`
Neumann János Talent Search Software Product Competition: `https://neumann.ibela.sulinet.hu/`
NT Syllabus: `https://github.com/niklaci/NT-Syllabus`

PENDroid High School Competition: `https://pendroid.uni-pen.hu/`
ProgRace Programming Contest: `https://prograce.hu/`
Programmers' National Dusza Árpád Memorial Competition: `https://isze.hu/dusza-arpad-orszagos-programozoi-emlekverseny/`
Robot Programming National Team Contest: `http://www.banyai-kkt.sulinet.hu/robotika/Robo-tverseny/robotverseny.html`
Robot Sumo Competition: `http://sagv.gyakg.u-szeged.hu/szumo/`
Talent Education Department of NJSzT: `http://tehetseg.inf.elte.hu/`
The Joy of Thinking Foundation: `https://agondolkodasorome.hu/en/`
World Robot Olympiad ™: `https://wro-association.org/`

L. Nikházy is a Ph.D. student at the Department of Media & Educational Informatics, Faculty of Informatics, Eötvös Loránd University in Hungary. His current research interest is computer programming talent education. He has been teaching mathematics in camps for gifted students since his early university years. After graduating from the Budapest University of Technology and Economics, and the Eötvös Loránd University, he started his career as a software engineer at Google. He gradually shifted to computer science education, and now devotes all his time to teaching talented children. He has been actively involved in organizing programming competitions and team coaching in Hungary in the past 3 years.

L. Zsakó Dr. is a professor at the Department of Media & Educational Informatics, Faculty of Informatics, Eötvös Loránd University in Hungary. Since 1990 he has been involved in organizing programming competitions in Hungary, including the CEOI. He has been the deputy leader of the Hungarian team at International Olympiads in Informatics 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.

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.IOInformatics.org/`

Hassinen, P., Elomaa, J., Ronkko, J., Halme, J., Hodju, P. (1999). *Neural Networks Tool – Nenet (Version 1.1)*.
`http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html`

Authors must submit electronic versions of manuscripts in PDF to the editors. The manuscripts should conform all the requirements above.

If a paper is accepted for publication, the authors will be asked for a computerprocessed text of the final version of the paper, supplemented with illustrations and tables, prepared as a Microsoft Word or LaTeX document. The illustrations are to be presented in TIF, WMF, BMP, PCX or PNG formats (the resolution of point graphics pictures is 300 dots per inch).

Contacts for communication

Valentina Dagienė
Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
Phone: +370 5 2109 732
Fax: +370 52 729 209
E-mail: valentina.dagiene@mif.vu.lt

Internet Address

All the information about the journal can be found at:

`https://ioinformatics.org/page/ioi-journal`

Olympiads in Informatics

Volume 14, 2020

Olympiads
in Informatics

Volume 14, 2020

9 771822 773007