TEAM COMPETITIONS ON PROGRAMMING FORMAT ACM ICPC

Sirojiddin Abdukarimov, Rakhimjon Shokirov

Abstract— this article is primarily intended for schoolchildren and students who are passionate about sports programming, who already have experience in personal competitions and are going to participate in team tournaments (or are already doing so). The reader is supposed to have an idea of how to solve various Olympiad problems. We hope that this article will be useful for teachers who prepare students for Olympiads of different levels, as well as for coaches of student teams. In this article I will say very little about what the Olympiad problems are and how to solve them. In order to solve problems quickly, efficiently and correctly you need to be a good programmer and mathematician, know a lot of algorithms and data structures from the sphere of computer science, have a great practical experience - a lot and regularly train. Moreover, after all, the ability to solve problems is to some extent a talent, a gift, because in fact, the solution of any complex Olympiad problem is always a creative process. You cannot come up with any general algorithm that would help you to find the right solution to an arbitrary problem.

Keywords: International Collegiate Programming Contest, Olympiad problems, sports programming, Association for Computing Machinery, International Olympiad in Informatics

INTRODUCTION

The International Collegiate Programming Contest (ICPC) is an annual, multi-tier competition held amongst college students on a global scale, with world championships every year. Last year alone, around fifty thousand students from three thousand universities participated in ICPC regional competitions. Because of its significant size involving a lot of talent and skillful people, multiple stakeholders are interested in the competition. Each of the competitions results in scoreboards, containing valuable data about the performance of teams. This data however is, up till now, never collected and stored in an open and free repository. The ICPC does keep track of the basic information such as teams' names and their final scores, but more detailed information has remained scattered across the internet. This paper describes the data collected and cleaned from the European, Latin-American, North American, South Pacific and World Finals from 2012 to 2018, opening up research opportunities for an in-depth look into the programming competitions [1]. Informatics Olympiads, like mathematics Olympiads, are widespread and have a fairly long history. ACM ICPC (Association for Computing Machinery International Collegiate Programming Contest) [2-4] has been held since 1977. IOI (International Olympiad in Informatics) has been held since 1989. These Olympiads allow revealing abilities both in mathematics and programming, and also ability to work under stress in a compressed time frame. These competitions are traditionally team events for students and personal competitions for schoolchildren. In Russia, computer science Olympiads for schoolchildren has a longer history. The book [5] contains all the problems of Moscow Olympiads in programming, which took place from 1980 to 1988. Materials for preparation for school Olympiads can also be found in books [6-8].

- *Sirojiddin Abdukarimov, State Inspection for supervision of quality in education under the Cabinet of Ministers, the Republic of Uzbekistan, Chief Specialist of the department of formation and testing of control materials, ss.abdukarimov@gmail.com, https://orcid.org/0000-0001-8993-8326.*
- *Rakhimjon Shokirov, PHD researcher, Tashkent State Technical University, raximvarresult@gmail.com , https://orcid.org/0000-0001-5378-9322.*

Differences between team competitions and individual competitions

Traditionally, in our country, as well as in the whole world, a more important role among the Olympiads for schoolchildren is played by personal competitions (IOI format), and for students - on the contrary, team competitions (ACM ICPC format). The main goals of the competition among students are already somewhat different. Students are older people, they have more practical experience and fundamental knowledge, as well as the ability to self-organize and cooperate. Therefore, in team competitions the level of complexity of tasks is already an order of magnitude higher, their number and duration of the competition is greater, and the requirements for solutions are tougher. In such conditions, one in the field - no longer "warrior", and in order to achieve results, you need a well-coordinated teamwork. Besides, when a team of students competes at international competitions, it not only protects the honor of its country and its city, but also to an equal extent its university. And here not just individual gifted persons compete, but various schools of Olympiad programming with many years of experience, united teams of professionals [2, 3].

Team participation.

Of course, the most important difference between team competitions and individual ones is the very fact that the participants of ICPC competitions solve the same tasks together, in a team of three people, not alone. This greatly complicates the team competition and gives the participants a lot of space to choose their tactics and strategy. When one participant speaks at the Olympiad, it is not so important for him in what order to read the terms of the problems. He himself, based on his individual abilities, decides which task he will solve first and which one he will postpone (knowing that one type of tasks is easier for him than the other), whether it is worthwhile to test the task he has just written again, or whether it is possible to start a new one, whether to look for an error in the solution by launching it with the help of a debugger, or by carefully rereading the source code of the program. He will call the variables in the code as he is used to, for nobody else but him will have to get into this code anymore and so on.

Number and level of tasks.

At student team competitions, participants are offered to solve six to twelve problems (instead of just three, as at school Olympiads). Moreover, the level of difficulty of most of these

problems is usually one order of magnitude higher than that of the schoolchildren's problems. It often happens that the idea of solving a problem breaks down into several parts touching on different theoretical areas, each of which can be quite complex individually (both in terms of the idea and in terms of implementation). And the conditions themselves are offered in English (which, although familiar to anyone who is somehow involved in programming, but still foreign). The student is required to have a higher level of knowledge and skills of finding solutions and their further implementation than the student. However, it is almost impossible to understand the conditions of each of the ten tasks alone (in a foreign language), find their solutions, take into account all the nuances, to predict possible errors (and if this was not done before the stage of implementation, then be able to find them and fix them). That is why in team competitions it is necessary to distribute tasks between team members and to cooperate in solving complex tasks.

The duration of the tour.

Competitions of students are held in one round, lasting five hours, without breaks (as opposed to school, where there are two rounds, which are held on different days). Five hours of uninterrupted hard work certainly require a lot of mental effort. At the end of the round, fatigue begins to affect the participants, new ideas for solving tasks are not so easy to think of, it becomes more difficult to find errors, attentiveness when writing code decreases, etc. All the more so because usually simpler tasks are solved at the initial stage of competitions and by the end of the round there are only "coffins ".

One computer for three people.

I want to emphasize an important nuance of the ICPC competition: although your team has three people, but the computer for three people you are given only one! What does it mean? Roughly speaking, since there are three of you, you can come up with solutions to the proposed tasks three times faster than if everyone solved the same set of tasks one by one, but you will be able to implement these solutions on your computer only at the same speed. The computer is the bottleneck of the team. And this critical resource must be used as efficiently as possible! In private competitions, time is the main resource. If a competitor is distracted by 10 minutes or ineffectively and ineffectively spends them trying to come up with a solution or to debug his program, he will lose exactly those 10 minutes (out of the total three hours of competition time). In team competitions the situation is more complicated. If one of the participants spends 10 minutes at the computer (for example, thinking through complex details in the process of algorithm implementation or trying to find an error in the debugger), then this time will be lost not only by him (even if he achieved a positive result for these 10 minutes), but also by the rest of the team members who could start writing other tasks in these ten minutes, the solutions of which they already know. In this way, a total of 30 man-minutes will be spent. This paragraph, in combination with the previous one, draws a very important and interesting conclusion. It can be formulated conditionally as follows: in individual competitions it is more important to test your already written decisions well for implementation errors before checking them by the jury (before the end of the tour), while in team competitions it is more important to make sure that the idea of your algorithm is correct and that it takes into account all possible cases, and that it always satisfies the restrictions on the resources used (time and memory). In personal competitions, it is not possible to send your decision for review again after it has not passed almost all the tests of the jury, except for a couple or three, because you, say, in one place by mistake wrote "i" instead of "j". At the same time, in team events, you will not get any benefit from writing a solution that delivers the right answer in 90% of the tests, or, for example, a solution that always works correctly, but on the highest allowed input data works for half a second longer than the limits set by the jury. This principle should be kept in mind, especially if you have already performed well in personal competitions but are only going to participate in team events.I will agree at once that this does not mean that in personal competitions it is not worth trying to think up the right algorithm instead of good heuristic. It also does not mean that when participating in team tournaments, you should not test your decision before sending it to the jury. Of course, sometimes you can limit yourself to testing only tests from a condition, if the task is very simple or the tour is coming to an end. Or if you have already properly tested all possible cases before, and now have found and corrected a specific error that does not affect the essence of the algorithm (an error in the data structure, a special case like n = 0, etc.), you can only check that the solution works on the cunning test, which it has not previously worked (and again on tests from the condition). However, it is almost always better to do a thorough testing. After all, I have never met a team in my life for which the problem was that they spend too much time on testing.

Interactivity testing.

Another important difference between personal competitions and team competitions is that in personal competitions no one has almost any information about the correctness or incorrectness of their decisions until the end of the tour. In recent years, during the tour is organized verification of the programs of participants only on tests from the conditions of the task. In team competitions such information is available to participants during the tour. In ACM ICPC competition, each solution is tested as soon as the team sends it to the jury and the result of the test is communicated to the team. The test is carried out in the following way: the solution is provided with test sets of input data (tests) prepared by the jury for the given task in turn. Each solution is run on each such set, and the corresponding output data it provides are checked for correctness. In addition, each time it is checked that the program execution has not caused an error and that it has met the limits in terms of memory size used and execution time. Such check is performed until the first test that the solution does not pass is met or until it successfully passes all the tests of the jury. In the first case, the number of the test, on which the solution "fell" is reported to the team together with the error type, and in the second case, the solution is counted as correct. Let me remind you of all possible types of messages that a team can receive as a result of sending a solution for checking (however, they are standard and used both in team and personal competitions):

- Accepted (OK) - The program has successfully passed all the tests.

- Compilation error (CE) - The program contains a syntactical error and the compiler cannot compile it correctly. This error occurs rarely and is usually caused by inattention of participants. It most often occurs when the wrong file name or

compiler type was specified by mistake when sending the solution.
- Time limit exceeded (TL) - The program on this test worked longer than the time limit (usually 2 s) and was interrupted by the testing system.
- Memory limit exceeded (ML) - At some point in time the program exceeded the allowed amount of memory used when solving this test.
- Runtime error (RE) - Execution time error. During execution the program generated an error and crashed (for example, the index exceeded the array boundaries, there was an attempt to process a null pointer, division by zero, etc.). The same result will be obtained if the return code takes a value different from zero at program exit.
- Security violation (SV) - A violation of security restrictions. This type of error is very similar to the previous one. It means that the program tried to perform some forbidden action, for example, to access someone else's memory area or try to create a file outside the current directory.
- Wrong answer (WA) - The solution worked through to the end, but the answer that came out of it is not correct.
- Presentation error (PE) - Incorrect output format. It is similar to the previous type, except that here the output format is already incorrect. For example, you needed to print an integer and the program printed a floating-point number, several numbers, an arbitrary line, created an empty file, or printed the answer to the wrong file.
All messages except the first two contain the number of the first test on which the error was detected. I should also note that in practice, due to the specifics of the checking program, error classes may often not be strictly observed, and, for example, instead of the PE error, the testing system may often produce a WA error, while the ML error in different systems may sometimes be produced as RE. Tests for a specific task are always run in the same predetermined order, the same for all. Usually a good set of tests for one task contains about 20-40 tests, but this is not a strict rule. Almost at all competitions, the first tests from the set of tests by the jury for a given task are tests from its condition, specified as examples (this is usually stated in the rules of the competition).

Availability of a monitor.
Another source of information about the course of the competition available to the participants of the team tournaments during the tour is an interactive monitor. The monitor is a table of results, which shows the positions of all teams at the current time, as well as much additional useful information. Each team is assigned a monitor line, which, in addition to its current position, indicates how many tasks this team has solved and what is the total penalty time. In addition, for each of the proposed tasks it is shown whether the team has passed this task, and if so, on what minute of the competition and at what attempt, and otherwise how many unsuccessful approaches have been taken by this team. In the first case, in the corresponding cell of the monitor there will be a record of "+X" type, where X is the number of unsuccessful attempts made by the team for this task (the total number of solutions sent minus one). For example, if the command has passed the task from the third attempt, the monitor cell will record "+2". Instead of "+0" it is usually written just "+". In the second case, in the corresponding cell there will be a record of "-X" type, where X is the number of unsuccessful approaches to the task. In both cases, the cell, along with the number of attempts at this problem, usually marks the time of sending the last solution for verification.

Recommendations.
To search for errors in the solution you should use printouts, not step-by-step execution (debug). It is an erroneous notion that you can easily find an error by using step-by-step execution of programs. Step-by-step execution is a painstaking process and takes a lot of time, and if the logic of the algorithm is quite complex, it consists of many parts, and it is not clear where the error is, it will be very, very difficult to catch the place of its first appearance. Besides, your computer will be busy all the time while you are debugging the program. If reading the printouts still does not give anything, then a good alternative to step-by-step execution may be to output intermediate data to the console right while the algorithm is running (so-called debug output). Its main advantage is that at the same time you can immediately see all the intermediate states of the program under test, not just the state at the current time. Besides, by adding the intermediate information output once, you can use it many times. At the same time, you should remember to comment it every time before sending the solution for checking if the program should output the main task response to the console, or if the algorithm is time-critical (because output of a large amount of information can significantly slow down the program). It is important that at the beginning of the tour ALL participants read the proposed tasks. One person is not always able to accurately assess the complexity of a problem and the probability to come up with a solution. While one participant may find the task too complicated and postpone it without even thinking about it, another participant may come up with an original idea that allows coming up with a fairly simple solution. Besides, one person may make a mistake, miss an important additional condition in the task condition, and misinterpret a phrase.

Conclusion.
Many aspects of Olympiad programming have not been considered in this article or have been described only briefly, but I hope that the given tips and ideas will help you to perform successfully in team competitions of different levels. In this paper the general principles of problem solving within the framework of the team student competitions on programming ACM ICPC format are considered. An attempt is made to formalize and describe the process of problem solving together with key aspects for each stage. The example is considered. Comments and recommendations are given. Of course, no instruction will replace real experience. Accordingly, reading these recommendations should not be an attempt to apply someone else's methods, but a meaningful perception of someone else's experience, set out in a set of recommendations.

REFERENCES
[1] de Boer, R. H., & de Campos, C. P. (2019). A retrospective overview of international collegiate programming contest data. Data in Brief, 25 doi:10.1016/j.dib.2019.104382
[2] IOI. International Olympiad in Informatics. Accessed at March 11, 2013, from http://www.ioinformatics.org; 2013.
[3] ACM ICPC. ACM International Collegiate Programming Contest. Accessed at March 11, 2013, from http://icpc.baylor.edu; 2013.
[4] Novandi, P. S. TOKI Learning Center - Sistem Pelatihan

Kompetisi Pemrograman Komputer. Bandung: Program Studi Teknik Informatika ITB; 2009.
[5] Moodle. (2013). Moodle Documentation. Accessed at March 11, 2013, from http://docs.Moodle.org/24/en/Main_page.
[6] Vane Spasov. Total LMS Documentation. Accessed at March 11, 2013, from http://TotalLMS.codeplex.com; 2012.
[7] Blackboard Inc. Blackboard: Technology and Solutions Built for Education. Accessed at March 11, 2013, from http://www.blackboard.com, 2013.
[8] Plateau LMS. Plateau LMS - eLearning Learning. Accessed at March 11, 2013, from http://www.elearninglearning.com/lms/plateau; 2012.
[9] Beth Davis, et.al. Moodle Moves to the Front of the LMS Adoption
[10] Berggren, J.L. (2003). Episodes in the Mathematics of Medieval Islam. Springer New York. ISBN 978-0-387-40605-3
[11] Knuth, D. (1980). Algorithms in Modern Mathematics and Computer Science. Stanford Department of Computer Science Report No. STAN-CS-80-786.
[12] Tabesh, Y. (2017). Mathematics Education in Iran from Ancient to Modern. In: M. El Tom & B. Vogeli (Eds.), Mathematics and its Teaching in the Muslim World. Series on Mathematical Education, Vol. 13, World Scientific.