Leonardo Gabriel Martínez Uribarren
MS. in Computer Science and Engineer
lgmu.cursos@hotmail.com

Required Readings

- Competitive Programmer's Handbook[1]. [Chapter 1. Pages: 3-16]
- The role of the University in computers, data processing, and related fields[2].
- Computing as a discipline[3].

1 Competitive Programming

Competitive Programming is an activity in which participants use a programming language to code specific logic and mathematical problems mainly focusing on getting the correct result, which means that the program have to finish and return a right output for a variety of inputs.

Each problem have a set of specifications that are used to score every coded program. Getting a high score when all specifications are accomplished. Such specifications can be:

- Finish the program on time
- Solve a problem under a specific lines of code
- The program should have a determined computational complexity
- Using just libraries made by participants

Moreover, to get a good place in competitions it is essential not only aiming to get an approved result but also in optimizing solutions for performance and efficiency, motivating participants to think critically about algorithm design, memory management, and computational complexity.

Problems can feature a wide range of topics such as graph theory, data structures, geometry, mathematics, bit manipulation, design, architecture, sorting data, etc. Where problems are designed to encourage participants to acquire a diverse computer knowledge and develop their problem solving skills.

1.1 Competitive Programming Basis

Competitive programming requires Computer Science and Engineering fundamentals in order to solve problems, where the former focuses on analysis and abstraction whereas the later on abstraction and design, both are essential areas to come up with a refined solution in terms of design, efficiency, implementation, and application. Because the Computer Science and Engineering studies the theory of algorithmic processes that analyze, describes, and transform information.[2][3].

Let's review some of the topics these disciplines study which are also used in contest to design problems:

Algorithm Theory	Study the efficiency and correctness of software. Verify if it is computable.
Data Structures	Study the representation of information in memory that can be static or dynamic. Described in a single data type or combined in a more complex group.
Programming Languages	Study the different methods and paradigms in which a programmer is able to give a series of ordered instructions to a computer.
Probability	Study the symbolic representation of a mathematical event and how likely are to occur.
Numerical analysis	Study the calculation of mathematical equations using numerical approximations obtained by an iterative process.
Databases	Study the methods to congruently store structured data.
Geometry	Study the space properties of figures and objects in different dimensions.

1.2 Advantages and Disadvantages

Advantages	
Enhancing Programming Skills	As well as any other activity like school, sports, arts, music, and even social skills, between many others can be improve by practicing. And competitive programming is not the exception, by studying and solving problems it is easy to mastery your ability at programming.
Preparing for Technical Interviews	High tech companies are aware of competitions and results, and even use competitive programming's rules, problems, and platforms to develop their own technical interviews.
Prizes and Rewards	By participate in competitions it is possible to get prestige and even rewards such as access to platforms, free subscriptions, goods or even money when get a good place.

Disadvantages	
Create bad programming habits	In order to do fast programming, it is common to avoid commenting the code, as well as not using descriptive variables names, and giving order to the project, because it is not necessary to maintain the code after competition is over.
Prioritize memorization over understanding	So as to improve the score on tournaments, it is common to start to memorize algorithms and blocks of code, and even prepare tons of code snippets instead of learning and understanding them.

1.3 How to start

1.3.1 Programming Languages

Having ample knowledge in at least one programming language is indispensable in order to solve problems. The following list shows some common languages used in competitive programming, where C++, Java, and Python are the most used in contests.

- C
- C++
- Java
- Python
- C#
- JavaScript
- TypeScript
- PHP
- Swift
- Kotlin
- Dart
- Go
- Ruby
- Scala
- Rust
- Racket
- Erlang
- Elixir
- Julia

1.3.2 Algorithm Theory

Algorithm Theory is one of the greatest pillars of Computer Science and all problems need using the knowledge field to be solved. Some of the common areas used in problems are[1]:

- Time complexity
- Memory
- Sorting
- Data structures
- Complete search
- Greedy algorithms
- Dynamic programming
- Divide and conquer
- Bit manipulation
- Graph algorithms
- Shortest paths
- Number theory
- Probability
- Game theory
- Geometry

1.3.3 Strategies

- To solve any problem it is convenient to follow the the following steps: Understanding the problem, Designing an algorithm and implementing the solution.
- Studying two or more programming languages is a good practice as certain problems are easy to solve in an specific language.
- Improving time management is useful because it takes up a lot of time and effort to get good at contests.
- Practice by solving problems is the only way to get better.

1.4 Platforms

Platform	URL	Contest	Training	Learning	Blog/Discussion
Codeforces	[**link**]				
Topcoder	[**link**]				
LeetCode	[**link**]				
CodeChef	[**link**]				
AtCoder	[**link**]				

References

[1] Antti Laaksonen. *Competitive Programmer's Handbook*. 2018, pp. 3–16. URL: https://cses.fi/book/book.pdf.

[2] Louis Fein. "The role of the University in computers, data processing, and related fields". In: *Commun. ACM* 2.9 (Sept. 1959), pp. 7–14. ISSN: 0001-0782. DOI: 10.1145/368424.368427. URL: https://doi.org/10.1145/368424.368427.

[3] P.J. Denning et al. "Computing as a discipline". In: *Computer* 22.2 (1989), pp. 63–70. DOI: 10.1109/2.19833.

[4] *Codeforces*. URL: https://codeforces.com/.

[5] *TopCoder*. URL: https://www.topcoder.com/.

[6] *LeetCode*. URL: https://leetcode.com/.

[7] *CodeChef*. URL: https://www.codechef.com/.

[8] *Hackerrank*. URL: https://www.hackerrank.com/.

[9] *atCoder*. URL: https://atcoder.jp/.