

Competitive Programming Beginners' Course

Coding Club, IIT Guwahati



WEEK - 1

What is CP and why should you do it?

- You are given a set of problems and you have to solve these problems in a fixed time.
- After solving, depending on your rank/percentile you will be awarded some virtual points.

Why CP?

- Mainly because it's competitive and Programming :)
- The Competitive part gives you a feel of sports and the Programming part might help you in your **future courses, internships and placements.**

PREREQUISITES

For this course there are no prerequisites as such. But to start CP you will need a decent level of knowledge of at least one programming language.

- General Order of Preference :-

C++ > Java > Python > Others

- It isn't that tough to switch between languages if you have a hold of at least one language.

**** We will assume that you are using C++ as your first language.**

SETTING UP AN IDE (Integrated Development Environment)

In simple words, an IDE is the environment in which you will be using to write your code, compile it and view the output. A suitable and well equipped IDE is very important, and not setting it up in the right way may cause trouble in the future.
(you may spend days fixing it).

Two of the most popular ones are :-

1. VS Code
 - a. [Windows](#)
 - b. [Mac](#)
2. [Sublime Text](#)

!! Don't mess up while setting it up otherwise you will easily waste days fixing it.

Where to learn C++?

What we are going to recommend is the fastest and the preferred way to learn C++. These are short videos , yet they are more than sufficient.

Note : **Do not waste much of your time learning the language (getting into tiny details).**

1. [Bucky's Tutorial](#) : Watch lectures 1 to 41 and 71 to 73.
2. Solve First 10-20 problems from [here](#) just to get comfortable with the language you chose.

Some websites from where you can learn and compete and some important competitions.

Websites :-

- **Codeforces :-** Most popular and widely used platform.
 - Contests are mainly of 6 types:- Div 3, Div 2, Div 1, Div 1+2, Educational Rounds , Global Rounds.
 - We recommend participating in all rounds **without worrying about rating changes**(ratings are temporary but experience is unique).
 - Do try to solve all the problems whose ratings are \leq your rating + 300. If you are not able to solve the problems, refer to their editorial if you still don't understand the solution, discuss it with your friends or ask your doubts in the competitive-programming-discussion channel on our Discord.
- **Atcoder :-** Educational and Weekly contests with quality problems.
 - 3 types of contests:- atcoder beginner/regular/grand contest
 - Must participate in Atcoder Beginner Contests(100 mins a week isn't that much)
- **Codechef :-**
 - 3 monthly contests :- Long challenge, Lunch-time and Cook-off
 - Weekly contests named starters

Resources :-

1. **Handbook (CPH)**:- You can find 90% of things you want to learn in this book, but I would not recommend reading this book completely and then starting solving problems. Just refer to the book when you come across a topic which you haven't heard before.
2. **CSES** :- The author of the Handbook has also made a compilation of 300 really good and conceptual problems which covers almost all CP topics. But it doesn't contain problems sorted according to difficulty so you will have to estimate it using solve count.
3. **USACO Guide**:- This is literally the "one stop shop". This website is generally referred to USA computing olympiad aspirants and it has everything in sorted order. And it has divisions like Bronze, Silver and Gold. It will mostly refer to the Handbook and explain solutions of CSES problems also.
4. **CP-Algorithms**:- Contains all CP topics but again not in sorted order of difficulty. So, refer to it only if you are stuck on a topic and couldn't find it in the Handbook.

Important Yearly Competitions :-

* We will remind you before all these contests when and how to participate.

1. Google Contests:- Don't miss any of them unless you have an exam at the same time.

a. Google Kickstart:-

- i. 8 times a year. All 8 contests are independent.
- ii. Sometimes the problems are irritating but still it's a google contest.

b. Google Codejam:-

- i. Held once a year. Probably the biggest CP contest.
- ii. 5 rounds in increasing order of difficulty, with elimination in each Round.
Prelims->Round A->Round B->Round C-> world finals

c. Google Hashcode:-

- i. Held once a year and is very different from normal CP contests.
- ii. You have to come up with a solution to a problem and improve it to gain more points. Problems couldn't be solved completely.

2. Facebook HackerCup:- We don't know what it will be called this year. It is also a once a year contest in topological order just like google codejam.

3. ACM ICPC:- Held once a year and the Biggest CP contest for college students. Also in topological order
Onlines -> Regionals -> World Finals

4. Codechef Snackdown:- Held once every two year. Next will be in 2023.

AFTER COMPLETING THE PREREQUISITES :-


After we have learnt a language, we can now get started with actual problem solving.

Problem solving requires understanding the problem statement and designing an effective algorithm that successfully solves the task at hand.

The efficiency of algorithms is **very** important in competitive programming. Usually, it is easy to design an algorithm that solves the problem slowly, but the real challenge is to invent a **fast algorithm**. If the algorithm is too slow, it will get only partial points or no points at all.

Thus comes into picture the concept of time complexity. The time complexity of an algorithm estimates how much time the algorithm will take for some input. The idea is to represent the efficiency as a function whose parameter is the size of the input. By calculating the time complexity, we can find out whether the algorithm is fast enough without implementing it.

Here are some resources which will help you understand time complexity :-

1. [CPH](#) (Competitive Programmers' Handbook) Chapter 2.
 - Do read this chapter thoroughly without skipping to get a complete understanding of Big O notation etc.
2.  [Big-O notation in 5 minutes – The basics](#)
3. [This](#) might help you in avoiding Time Limit Exceeded error

Following are some challenging problems based mainly on implementation :-

- [Increasing Array](#) ([Solution](#))
- [Permutations](#) ([Solution](#))
- [Number Spiral](#) ([Solution](#))

- [Chloe and the sequence](#) ([Solution](#))
- [Vikas and Squares](#) ([Solution](#))
- [Trailing Zeroes](#) ([Solution](#))
- [Masha and two friends](#) ([Solution](#))

*Try to attempt a question for a minimum of 30 minutes before checking the solution.

*We have attached the codes with solutions. If you still have any doubts please ask us in the discord channel !