# Competitive Programming L1

## Content

- Why competitive programming
- Why are algorithms and data structures are important?
- Big-O,Big-theta and Big-Omega notations
- Problem format
  - Problem statement
  - Input
  - Output
  - Estimating if a problem will run on our coding style
  - Estimated complexity we require to build solution
- Simple Questions i.e. very easy from hackerearth, cakewalk or beginner from codechef and div2 A from codeforces
- Algorithm paradigms
  - Brute force
  - Divide and Conquer
  - Greedy
  - Dynamic Programming

## Why Competitive Programming?

1. **Helps learn programming deeply**
   It is common practice to learn a concept this way. First use a library, and if interested in it, then go deeper into a topic, implement different processes of library yourself so get better understanding of the behind the scene work of an algorithm/process. But note **implementation** part, one needs to solve various problems to get good in implementation of various ideas in mind to code.
2. **Asked in many interviews**
   Given an hour, what is the best way to judge if a programmer who has not worked professionally ever is good enough for a job or not. Which leaves most interview questions to
   - Syllabus Questions
   - Basic Implementation
   - Easy problem solving
   - Medium problem solving
3. **Gives a platform to prove yourself**
   Many people sometimes do not get opportunity to proves how good of coder they are, sometime

companies of interest do not show up at campus placements, and sometimes we do not have an inside reference to get our resume up the stack.
Many companies like Google, Microsoft, Directi, Codenations etc recruit people on the basis of their competitive programming performance or they specially organise a contest for such purpose.
For example: Google organises Kickstart
and CodeNations organise CodeAgon
and there are many more examples

4. **It is really really fun**
   The moment when you are brainstorming for 2-3 days, finally an idea clicks and you just know that it is going to be an AC(Accepted), those green results are the best moment of your life.
   Competitive Programming is the best sport for a programmer(well my opinion though)

# Why are algorithms important?

This is a very important question before starting to learn anything. Why should we learn anything at all?

To explain this, I would directly speak what the bible (CLRS) taught me.

There are two algorithms

1. Merge Sort
2. Insertion sort

Time complexities of both are $O(nlog(n))$ and $O(n^2)$.
Let's say we have two computers A and B. A is $1000$ times faster than B,
For a database with $10^6$ entries, Merge Sort would need around $10^7$ operations while Insertion sort would need $10^{12}$ operations, this means that even if you run Merge Sort on computer B and Insertion Sort on Computer A, Computer A will take $100$ times the time taken by computer B to sort them.

P.S. Do not ask why we need sorting at all, this is what this whole SIG is about.

# Big-O, theta and Omega notations

Well let's get into geeky part of competitive programming, but you know its fun to later use this

```
for(int i = 0; i < n; ++i){
    line 1
    line 2
    line 3
}
```

if we notice we will see that line 1,2,3 are repeated $n$ times, if time execution are as follows

| Line | time |
|---|---|
| 1 | $t_1$ |
| 2 | $t_2$ |
| 3 | $t_3$ |

Thus total time $T$ = $nt_1 + nt_2 + nt_3$
$T$ = $n(t_1 + t_2 + t_3)$
since $t_1 + t_2 + t_3$ is constant
Therefore time is proportional to only $n$
Therefore, we say that time complexity of that code snippet is $O(n)$ or $\theta(n)$ or $\Omega(n)$ (Will learn difference in a few paragraphs (you see what I did here, I don't know if you will read in next minute or later so I said in a few paragraph, Dammit I am tired of typing I will just paste from geeksforgeeks))

Similarly, in this snippet

```
for(int i = 0; i < n; ++i){
    line 1
    for(int j = 0; j < n; ++j){
        line 2
        line 3
        line 4
    }
}
```

| Line | time |
|---|---|
| 1 | $t_1$ |
| 2 | $t_2$ |
| 3 | $t_3$ |
| 4 | $t_4$ |

This make equation
$T = nt_1 + n^2t_2 + n^2t_3 + n^2t_4$
As we know that $n^2$ gives more appropriate sense of proportionality rather than $n$
Therefore time complexity get to be $O(n^2)$ or $\theta(n^2)$ or $\Omega(n^2)$

## Difference Between Three Notations

1. **Big-O represents worst case complexity**:
   Let's consider the example of an algorithm running and there is always an upper number of calculations that will happen in an algorithm.
   For example in finding a subset with certain property from a set of size $n$, there will be at max $2^n - 1$ operations are that is the total number of possible set
   That makes Worst case complexity of this algorithm to be $O(2^n)$
2. **Big-theta represents average case complexity**:
   We are skipping this now, because it will get confusing. Will come back to this later.
3. **Big-Omega represents best case complexity**:
   Kind of pretty useless, but can be used to see why selection sort is very very useless algorithm. LOL

# Competitive Programming Problem Sample

Some sample problem from famous competitive programming sites(not to be solved, well unless you want to but don't ask me till we get there)
Codeforces
CodeChef
HackerRank
HackerEarth

You will notice that all these problems are kind of similar in format which is

1. Problem statement
   This is the details of problem. Sometimes it is straight forward, sometimes we have to understand this in order to plan the algorithm or data structure to be further used.
2. Input
   This tells the input format in which input has to be provided, there can be no extra "Please input a number", Don't make this mistake please. just please.
3. Output format
   Output to all problems must be given in the exact format asked by the problem setter.
   Sometimes including newline(if not asked) or not using a newline(when asked) can lead to WA(Wrong Answer)
4. **Constraints**
   Perhaps the most important part. I will tell you in class. Its hard to explain via text.

This is a little important
**We will be coding in C++, so if you are not familiar with it, you must either learn it or just learn algorithm in class and try implementation your Java Code your self**

If you are familiar with C but not C++.
Just remember this for now(Believe C++ will be better here than C)

```
#include<bits/stdc+.h>
```

Instead of

```
scanf("format string", parameters)
printf("format string", parameters)
```

Use

```
cin >> var1 >>  var2;
cout << var1 << var2;
```

And please do not forget to use **.cpp** extension instead of **.c**. I used to forget, dumbest thing I ever did.

Questions to practice

1. Hello
2. Birthday Cake Candles*
3. Mini-Max Sum*
4. Staircase
5. Plus-Minus
6. Diagonal Difference
7. A Very Big Sum*
8. Factorial
9. Count Divisors*
10. Prime Number*
11. Prime Numbers***(really good)
12. Anagram (will use in hashing as well)
13. Find Product(Modular Arithmetic)
14. Seating Arrangement(Practice Implementation)
15. Sherlock and XOR
16. Rajat and Odd Frequencies
17. Young Physicist
18. Beautiful Matrix
19. Queue at School
20. Lights Out
21. Coding Rank
22. Coding Game(Binary Search)
23. Planning the expedition (Binary Search)
24. Stages
25. Piles With Stones

26. Points in Segment
27. Segment Occurrances
28. New Building for SIS
29. Badge
30. Pallindromic Twist
31. Find Square
32. Unnatural Conditions
33. Packets
34. Reach Median
35. Non-CoPrime Partition(Maths)
36. Cover Points
37. Mishka and Contest
38. Chef and Polygon Cake
39. Spell Bob
40. Chef and Eid
41. Split Stones