# DSA

A primer for competitive coding

## Complexity and Constraints

- You should assume a computer can do about $10^8$ calculations per second
- Check how much time is given for your program to run
- Check the values of the constraints to determine the complexity of your code

For example -

If there is a constraint $1 < n < 10^5$
Then the complexity of your algorithm should be at least O(n*log(n))

If $1 < n < 10^{10}$ then the complexity should be at least O(log(n))

## Problem Types

Different programs will often require knowledge of different programming paradigms
The types of *basic* problems you'll be expected to solve are

- Ad - hoc
- Complete Search
- Divide and Conquer
- Greedy
- Dynamic Programming

There are other types of questions that might require knowledge of graph theory, computational geometry or game theory but they will be covered in later sessions

## Ad-hoc Problems

- These problems tend to be the easiest
- They require no special algorithmic paradigm
- Questions are straightforward and logical
- However, corner cases and edge cases might make them difficult to solve

## Complete Search

- Problems that require some sort of iterative algorithm
- Array and String manipulations are commonly asked
- Complexity should be around O(n)
- Some Backtracking questions may also be asked here

## Divide and Conquer

- The idea is a problem can be divided into multiple subproblems that can be solved recursively
- Example - Binary Search
- Usually these problems expect knowledge of Binary Search and sorting algorithms like Quick Sort and Merge Sort
- Complexity of the algorithms will usually have log in them
- Binary Search Complexity is O(log(n))
- Merge Sort and Quick Sort Complexity is O(n*log(n))

## Greedy Problems

- Here, the algorithm will try to calculate the local optimum during each stage of the algorithm to reach the global optimum
- In the real world, they are not very helpful because local optimum almost never leads to global optimum
- However, some competitive questions are designed to be solved by greedy algorithms
- When they work, greedy algorithms can reach the optima very quickly

## Dynamic Programming

- Solving DP problems are some of the hardest especially for beginners
- DP is solving problems by dividing them into sub problems
- It is different from Divide and Conquer because here we are precomputing values to avoid the overlapping subproblems problem
- DP problems are of two types, top-down and bottom-up
- The top-down approach is also called memoization