Coding Club
IIT Guwahati

CP.begin()

Coding Club, IIT Guwahati

Week - 3

Binary Search :-

This is one of the most useful and important algorithms you will ever come across. It is used to solve problems which have some **monotonic features**.

The main idea of the algorithm is to **divide the problem into half at each stage**.
Let's take an example of a classical problem. Suppose you are given an array of elements whose elements are sorted in increasing order and you are asked the smallest number greater than k which is present in the array.

Now take some time and think of some strategies.

One possible strategy can be:
Iterate over the array from 1 to N, once you get to a position whose value is greater than k then this is the answer.

Optimal Strategy: Binary search
You can divide the array into two halves **1 to N/2** and **N/2 to N**. Now let's check in which half is our answer present. If it's in the left half then a[N/2] has to be greater than k. If not, then it has to be in the right half.

Thus, just by checking the value of a[N/2] we can get rid of one half.

An important point to note is that after every query the length of the array in which we search becomes half.
So **N -> N/2 -> N/4 -> N/8 ..... 1** , and we only need to ask log(N) queries.

So, Binary search is only this much (just think about removing one half).

Here are some great resources which will teach you how to think about binary search and how to implement it:-

- Binary Search tutorial (C++ and Python)
- All 5 steps of codeforces edu Courses - Codeforces

STL stuff related to binary search:

In almost all Containers in C++ you can use inbuilt binary search which works in O(log(n)).

These two you will use most frequently.

- Vectors :- You need to first sort the vector for these to work correctly.
  - Binary Search
  - Lower Bound

    - Upper Bound
- Sets / Multisets :-
    - Find (works like binary search)
    - Lower Bound
    - Upper Bound

Note that both have different syntax. If you use the syntax for vectors in Set then its time complexity won't be O(logn), it will be O(n), and will probably give TLE. You have to be very careful about the syntax.

Problems:-

- Problem 1
- You can attempt all the problems attached with codeforces edu binary search.

Some additional problems :-

- Problem 1
- Problem 2
- Problem 3
- Problem 4
- Problem 5
- Problem 6
- Problem 7
- Problem 8
- Problem 9 (Tough)

More Problems:-

In week 2 you learnt Binary exponentiation. You can relate it to binary search also. First try to write a code yourself to calculate a to the power of b. If you can't come up with a solution, you may refer to this video Binary Exponentiation. I advise you to devote, at the very least, half an hour before watching the video.

Ternary Search :-

It is also a search algorithm like binary search. You can solve ternary search problems using binary search also but that requires more observation and not so clean implementation. Ternary Search can be applied on sequences which have exactly one global minima or exactly one global maxima.

Tutorial:-

- Ternary Search Tutorials & Notes | Algorithms | HackerEarth
- Ternary Search - Algorithms for Competitive Programming

Problems:-

- Problem 1
- Problem 2
- Problem 3
- Problem 4 (Can be solved using maths but do it using ternary search)

Two Pointers :-

The name "Two Pointer" suggests the use of two different pointers, and it is exactly that, but without actually using C++ pointers (Well, you **can** use, but why would you want to?). It's just a fancy name for a technique which uses two variables in a single loop. The variables are generally used to keep track of indices of an array or string, and generally in a sorted array.

For example, suppose you are given a sorted array A and asked to find out a pair of numbers such that their sum is equal to X.

One obvious way is to solve the problem using two for-loops. The time complexity will be O(N^2) in the worst case. But the same problem can be done in O(N) complexity using double pointers. Suppose the two variables are U and V. U initially points to the first index of X, and V to the last index. If currently A[U]+A[V] < X, we can safely increase U, since it is guaranteed that it will increase the value of A[U]. Similarly. if the sum is more than X, we can decrease V. If the sum is equal to X, we have reached the solution.

Remember, it's not necessary that one pointer should be at the start of an array, and the other at the end. It is entirely possible that both of the indices start from the beginning, but the idea remains largely the same. You just have to check which index has to move, and in which direction, to move the result closer to the required value.
For example, if the above problem had asked for the difference instead of the sum, you would have to start both indices from the start, increase V if the current difference is less than X, and increase U if it is greater than X.

Try to write the code for the above problems after going through at least step 1 of codeforces edu.

Resources:-

- All steps of codeforces edu - Courses - Codeforces
- CPH

Problems:- Try to attempt all problems on codeforces edu.

Additional Exercise:- Many of the two pointer problems can be solved using binary search as well. Here are some problems which you have to solve using binary search as well:-

- Problem 0 (discussed above)
- Problem 1
- Problem 2
- Problem 3
- Problem 4
- Problem 5
- Problem 6

Bit Operations :-

In Competitive Programming or Programming in general, writing numbers in their binary form might help you. It's really necessary to know different properties of binary representation of numbers and how to perform binary operations.

Few resources:-

- CPH (pg 96 to 99)
- **Bitwise Operations tutorial #1 | XOR, Shift, Subsets**
- **C++ Bitsets in Competitive Programming**

Problems:-

- Read it and solve the problems Part 1 Part 2 (This blog is the written version of video)

Introduction to Dynamic Programming :-

No doubt this is the widest topic in Competitive Programming, but it's not something which cannot be taught.

Few resources:-

- CPH (pg 65 to 69) (If you don't understand in one read, read it again)
- Errichto (Lecture 1 and Lecture 2)

If you didn't understand even after referring to the resources. Don't worry it's normal, just jump to solving problems.

Problems:-

- Problem 1
- Problem 2
- Problem 3
- Problem 3
- Problem 4
- Problem 5
- Problem 6 (Knapsack Problem, Very Important, try it first if you have no clue how to solve it then refer this or this)

More Problems:-

- Problem 1
- Problem 2
- Problem 3
- Problem 4
- Problem 5

*In future if you want to study more things in DP then you can refer Atcoder DP Contest(It covers almost all DP techniques) and for reading more USACO Guide

Note for Problem 6 in Binary Search Section :-

- If you haven't tried it yet, stop reading from this point onwards.
- If you are still reading, you should have devoted, at the very least, half an hour on the problem. If not, try to solve it sincerely first.
- You will not be able to solve Problem 6 with Binary Search. You will have to use the set data structure for solving it. You should still try solving it using Binary search though, since it is important to learn about its limitations.