Competitive Programming (SoC'25)

Project id - 22

Mentor - Himanshu Shete(23B0770)

## Week 1 : Arrays, Time Complexity, ***Binary Search***, Two Pointers, Sorting

### Theory:

(these are just resources you can always learn from youtube or other sources)

1. Arrays:
   1. <https://www.geeksforgeeks.org/introduction-to-arrays-data-structure-and-algorithm-tutorials/>
   2. <https://www.geeksforgeeks.org/array-subarray-subsequence-and-subset/>
2. Time Complexity
   1. <https://www.w3schools.com/dsa/dsa_timecomplexity_theory.php>
3. ***Binary Search***
   1. <https://www.youtube.com/watch?v=GU7DpgHINWQ>
   2. <https://cp-algorithms.com/num_methods/binary_search.html>
   3. Stl: binary\_search(), lower\_bound(),upper\_bound(): <https://www.geeksforgeeks.org/binary-search-functions-in-c-stl-binary_search-lower_bound-and-upper_bound/>
4. Two Pointers
   1. <https://www.geeksforgeeks.org/two-pointers-technique/>
5. Sorting
   1. <https://www.geeksforgeeks.org/sort-c-stl/>
   2. Visualize:- <https://visualgo.net/en/sorting>
   3. <https://www.w3schools.com/dsa/dsa_algo_bubblesort.php>
   4. <https://www.w3schools.com/dsa/dsa_algo_selectionsort.php>
   5. <https://www.w3schools.com/dsa/dsa_algo_insertionsort.php>
   6. <https://www.w3schools.com/dsa/dsa_algo_quicksort.php>
   7. <https://www.w3schools.com/dsa/dsa_algo_countingsort.php>
   8. <https://www.w3schools.com/dsa/dsa_algo_radixsort.php>
   9. <https://www.w3schools.com/dsa/dsa_algo_mergesort.php>

### Problems:

(increasing difficulty, maintain a git repo)

1. Arrays and Sorting:
   1. <https://leetcode.com/problems/two-sum/description/?envType=problem-list-v2&envId=array> (you can do it in o(n2))
   2. <https://www.codechef.com/practice/course/arrays-strings-sorting/INTARR01/problems/DOMINANT2>
   3. <https://www.codechef.com/practice/course/arrays-strings-sorting/INTARR01/problems/DISTINCTCOL>
   4. <https://cses.fi/problemset/task/1094>
   5. <https://www.codechef.com/practice/course/arrays-strings-sorting/INTARR01/problems/EQUALELE>
   6. <https://www.codechef.com/practice/course/arrays-strings-sorting/INTARR01/problems/CFRTEST?tab=statement>
   7. <https://codeforces.com/problemset/problem/300/A>
   8. <https://www.codechef.com/practice/course/arrays-strings-sorting/INTARR01/problems/MISSP?tab=statement> (bit operations needed else tle)
2. Time Complexity: (Practice observing complexity via performance on large testcases)
3. Binary Search: (i would suggest implimenting binary search every time instead of using std::binary\_search() of cpp stl)
   1. <https://leetcode.com/problems/binary-search/description/>
   2. <https://www.codechef.com/practice/course/binary-search/INTBINS01/problems/TRICOIN>
   3. <https://www.codechef.com/practice/course/binary-search/INTBINS01/problems/WAV2>
   4. <https://www.codechef.com/practice/course/binary-search/INTBINS01/problems/SHKNUM>
   5. <https://www.codechef.com/practice/course/binary-search/INTBINS01/problems/MINEAT> (\*)(this problem is beautiful go through the solution)(calculate at only the binary search locations instead of calculating everythin then binary search)
   6. <https://www.codechef.com/practice/course/binary-search/INTBINS01/problems/SNAKEEAT> (\*\*)
4. Two Pointers and Sliding widow (practice more)
   1. <https://www.codechef.com/practice/course/two-pointers/POINTERF/problems/PREP69?tab=statement> (easy)
   2. <https://www.codechef.com/practice/course/two-pointers/POINTERF/problems/PROC18A> (classic)
   3. <https://www.codechef.com/practice/course/two-pointers/POINTERF/problems/PREP68> (another classic)
   4. <https://www.codechef.com/practice/course/two-pointers/POINTERF/problems/PREP17>
   5. <https://leetcode.com/problems/longest-palindromic-substring/description/?envType=problem-list-v2&envId=two-pointers>
   6. <https://leetcode.com/problems/container-with-most-water/description/?envType=problem-list-v2&envId=two-pointers>
   7. <https://leetcode.com/problems/3sum/description/?envType=problem-list-v2&envId=two-pointers> (threesome)
   8. <https://leetcode.com/problems/4sum/description/?envType=problem-list-v2&envId=two-pointers> (foursome or k-sum)