Part - 2

In this week, we'll look at sorting, searching algorithms and divide and conquer techniques. From the book it is chapter 3 and some parts of chapter 5. This week's topics are less so that you can revise over the data structures from the last week and get hold of these data structures.

💡 Go through the chapters 3 and 5 in the book (sent in the group) after you have gone through the links below.

Sorting

Sorting algorithms are essential tools in computer science, and in this week we will study some of the most popular ones, including selection sort, insertion sort, and merge sort. We will also learn about divide and conquer techniques, which are used to break down a problem into smaller subproblems, solve them separately, and then combine the solutions to obtain the final result.

Check out this GeeksforGeeks article for sorting algorithms. You can visit all the algorithms once if you wish, but the important ones are selection sort, bubble sort, merge sort, insertion sort, quick sort and heap sort.

Mostly during problems, we use direct STL functions, but it is important to know the backgrounds of these algorithms as a slight twist in the question can make the STL function fail. This is the link to STL sort function.

Try this: Also search what happens if you sort the vector of pairs, or vector of vectors and How to customise the sorting process. Say that you want to sort using the sorting function in C++ STL but you have to sort it in decreasing order or some of other criteria. (By now, you should have figured that sorting using C++ STL will sort in increasing order).

Searching Algorithms:

Searching algorithms are used to find the location of a particular element in a given data structure such as an array or a tree. Here are some commonly used searching algorithms:

1. Linear Search: This algorithm checks each element of the array or list until the target element is found. It is a simple and straightforward algorithm but not very efficient for large data sets.

   Here's a link to the linear search algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/linear-search/

2. Binary Search: This algorithm is used for searching in a sorted array by repeatedly dividing the search interval in half. It is a more efficient algorithm than linear search for large data sets.

   Here's a link to the binary search algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/binary-search/

3. Interpolation Search: This algorithm is used to search for a specific value in a sorted array that is uniformly distributed. It uses the position of the target element to estimate its location.

   Here's a link to the interpolation search algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/interpolation-search/

So, here is a quick question: which is better on an array which is not sorted? linear search, or sorting followed by binary search. (Hint: think of time complexity)

Divide and Conquer Algorithms:

Divide and conquer algorithms are a class of algorithms that break down a problem into smaller sub-problems, solve them recursively, and combine the solutions to solve the original problem. This is actually a very important design idea. Here are some commonly used Divide and conquer algorithms:

1. Merge Sort: This algorithm sorts an array by dividing it into two halves, sorting the two halves recursively, and merging the sorted halves.

   Here's a link to the merge sort algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/merge-sort/

2. Quick Sort: This algorithm sorts an array by partitioning it into two parts, one with elements smaller than a chosen pivot and the other with elements greater than the pivot, and then recursively sorting the two parts.

Here's a link to the quick sort algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/quick-sort/

3. Binary Search Tree: This algorithm creates a binary tree in which the left subtree contains nodes with values less than the root node and the right subtree contains nodes with values greater than the root node. This allows for efficient searching, insertion, and deletion of elements.

   Here's a link to the binary search tree algorithm on GeeksforGeeks: https://www.geeksforgeeks.org/binary-search-tree-data-structure/

Merge sort and Quick sort are usually not required to be implemented by us for CP. But the crux of learning from that is for understanding what is divide and conquer idea. So, Let’s see few a difficult question that are solved easily by the divide and conquer idea.

Non-dominating set of points: In a set of points over 2D plane, among 2 points (x,y) and (a,b) in the set, if x > a and y > b, then (x,y) is said to be dominating (a,b). A point is said to be non-dominating, if there is no other point in the set which dominates it. Now, given a set of points in first quadrant that is both the x and y coordinates are positive, how will find the non-dominating set of points from that set. (Hint: There are two ways of course, one to use divide and conquer or to find other way around without using it. But try to get an Algorithm which uses divide and conquer idea).