Algo 2020 - Homework 5 - PART 1

Question 1 (C++ code required) (2.5*4=10) points)

a. You are given an array A of N integers, and a number X, Write an algorithm with complexity of O(N log N) which should be able to find a pair **Pi, Pj** of values within A such that **Pi+Pj = X**.

b. Now You are given a sorted array A of N integers, and a number X, Write an algorithm with complexity of O(N) which should be able to find a pair Pi, Pj of values within A such that **Pi+Pj = X**.

c. Write an **O(N log N)** algorithm which should detect whether the sets (two vectors having all distinct values within a vector) A and B overlap or not. Your program should return there are how many elements common between two sets.

d. Now write an **O(N)** algorithm for Part 3, if the given two sets are sorted.

Question 2 (5+5 points)

1. Given an array of **N** elements, write an algorithm which should find the minimum and 2nd minimum in exactly or less than **N + log N** comparisons. Finding it in exact **2N** comparison is obvious and carries no credit.
   HINT: Think of it as the binary tree created in such that at the lowest level there are all the elements and then on one level up there are all the elements which are selected by comparing two consecutive elements and moving the element forward which is minimum of the two. So on the lowest level there will be N elements and then one level up there will be N/2 elements and so on. On the top most level where there will be one one element that will be our minimum. Now observe that tree and see where can the 2nd minimum can? (C++ code NOT required)

2. You have N elements array but you are given that there are only K distinct values which has been repeated again and again in any order. For example N could be 1000 and K could be 10 means that the array contains only 10 values which are being duplicated anywhere in the array. Now you want to sort that array. You must have to give appropriate implementation using **`vector, map or set`** (libraries in STL) or anything you like. Your algorithm should take **O(N log K)** time instead of **O(N log N)**. Drive time complexity.
   HINT: Try to use any balanced tree. (C++ code required)

Question 3 (C++ code required (3+3+4 points)

You are given a sorted array of **N** elements. But unfortunately the array has been circularly shifted K times and you don't know that K. Design the following algorithm:

1. Detect How many times the array has been circularly shifted. Your Algorithm should run in **O(log N)** times.
2. Find the Maximum element in the array. Your Algorithm should run within **log N** bound.
3. Design an **O(N)** algorithm which should undo those rotations in such a way that now the array is sorted again.

Question 4 (C++ code required) (2+2+6+5 points)

Code Investing in the Stock Exchange problem:

- **$O(N^3)$ Algo and $O(N^2)$ Algo O(N log N) Algo**
- Design an **O(N)** algorithm, MY PF class had solve this problem in class, so I guess you can also do it (though we will cover it after mid term in the topic under dynamic programming). (5 points)
  HINT: Use the following ideas to develop a nonrecursive, linear-time algorithm for the maximum-subarray problem. Start at the left end of the array, and progress toward the right, keeping track of the maximum subarray seen so far. Knowing a maximum subarray A[1..j], extend the answer to find a maximum subarray ending at index j+1 by using the following observation: a maximum subarray A[i..j+1], for some 1≤i≤j+1. Determine a maximum subarray of the form A[i..j+1] in constant time based on knowing a maximum subarray ending at index j.

Question 5 (C++ code required) (3+3+3+3+3=15)

Consider two sums, $X = x_1 + x_2 + x_3 + \ldots + x_n$ and $Y = y_1 + y_2 + y_3 + \ldots + y_m$. Give an algorithm that finds indices i and j such that swapping xi with yj makes the two sums equal, that is, $X - x_i + y_j = Y - y_j + x_i$, if they exist. Analyze your algorithm. (You can use sorting as a subroutine, by calling Sort function of STL).

Question 6 (C++ code required) (10 points)

Write down the code of Multiplication of two N digits numbers **A x B**(stored in the form of base 10 numbers, it will be a vector of integer where each value will hold one digit of the number).

1. Write down the code of our high school multiplication algorithm (this should be two nested loops, i believe).
2. Write down the code for Divide and Conquer algorithm for multiplication by using 4 recursive calls.
3. Write down the code for Divide and Conquer algorithm for multiplication by using 3 recursive calls, gauss method. (in DPV you will find the solution for base 2 program).
4. Using FAST addition find out the multiplication of **A x B**.
5. COMPARE ALL THE ABOVE 4 ALGORITHMs by TAKING randomly generated 10 thousand digits long numbers and compute multiplications.

Question 7 (C++ code required) (15 points)

(READING QUESTION): Read Strassen's algorithm for matrix multiplication, First write its O(N^3) implementation (both iterative and recursive explained in the book). And then read from CLRS, its Divide and Conquer recursive formulation and make the algorithm for better implementation. Test your program by taking a 128x128 , 256x256, 1024 x 1024, 1MBx1MB random matrix and see how the two algorithms perform. Analyze the time complexity of Strassen algorithm.

Question 8 (C++ code required) (10+10 points)

Implement the following:

1. Implement Modified Randomized QuickSort
    a. Randomized Partition algorithm yet again (which we did in class)
    b. Implement the Randomized Select Code (i.e Find the k'th smallest element in an array), the entire code is given in the lecture slides (of lecture 12).
    c. Now modify QuickSort implementation, instead of calling partition in the QuickSort call RandomizedSelect which must given the exact middle element and also partition the array into exact two halves.
2. Implement Modified Randomized QuickSort
    a. Now write the code for DETERMINISTIC-SELECT as explained in the slides (by taking groups of 5 and using the idea of prune and search Median of Median Searching).
    b. Implement Deterministic-QuickSort which uses DETERMINISTIC-SELECT.
3. **TEST ON HUGE DATA (like 50MB)**

Question # 9 (It has three parts)

The Challenge 1* (pseudo code required) (20 points)

(This is the question of HW2 - Most of the class did not solve this problem that is why I am giving this question again).
MAKE THE RECURRENCE OF THE FOLLOWING PROGRAM AND SOLVE IT for N people and more than half are Trustworthy people and the rest are fakers.

You wake up after your Titanic sank, and you find yourself on the shore of the sea, an island. Beside everything, what an island has (its natural beauty), there are tribesmen there too. You come to know that that every tribesman of this island is either faker(he may or may not lie) or trustworthy(always say the truth). You can ask them questions like who is a faker. And who is trustworthy? There are total of 100 people in this island. And you can ask as many questions as you want. The only information known is, that number of Trustworthy people is greater than fakers. Any answer having greater than 500 questions is not worthy enough to be considered. Figure out who is a faker and who is a trustworthy person?

Hint:

You have to find at least one trustworthy by whom you can ask who is a trustworthy or lier. The strategy is inspired from the beautiful Geometric series. One idea could have been to take one person out and ask from everyone "Is he the trustworthy one?" if majority say yes then we have found the one trustworthy and we may ask him about everyone. But the issue is you could be unlucky in that case you have to choose the 2nd person and repeat that would be a very bad solution. It will work but in that case you may have to almost select 49 bad choices of fakers.

Now the idea is make pair of 2 each (resulting 50 pairs i.e. N/2 pairs) and just ask those pairs about each other Here could be the possible answers (F for faker, T for Trustworthy).

Case 1	**F**	**F**
Case 2	**T**	**F**
Case 3	**T**	**T**

In all those cases where cases 1 and 2 arise we reject the pairs. In case 3 we choose one representative and ask the other person in the pair to sit down. With this one process greater than half of the people have been pruned and you should prove that "the Trustworthy-men in majority" condition will still be valid (you should prove that?). With this one step we have asked 100 questions but the searching space have reduced to half, now we repeat the same process of shrinking and on the worst case the number of questions will look like the following series
100 + 50 + 25 + … + 1 that will always be less than 200 questions. Hence we have found one Trustworthy in around 200 questions.

Be careful that with this procedure it may be possible that there are odd number of people remaining on a certain step what should we do in that case? We have to think about that. Hint is in incorporating that issue we might have to do some extra steps which might take the number of questions close to 300.

The Challenge 2: Facebook wants to hire you, and they have one interview question? (15 points)

One of the facebook server has been crashed, it has stored millions of users accounts images data (stored in the form of folder, where each folder contains N images uploaded by the user as profile photo, **where each photo may not be his own photo, but in Most Of The CASES the majority of the user images collection contains user's photo**). In the interview they have asked you to design the following algorithm.

- ➢ You are given a subroutine **Equal_images(image A, image B)** it returns true if both images contain the same person.

A. Given a list/array of N images, design an O(N) time algorithm which finds out the **Majority image**. Note that you cannot compare the two images (like < or >), other than calling **Equal_images** function which just tells whether A and B contains the same person.
Note: What is Majority Image? It is an image(containing a human face) which is present in more than half of the collection of images.

B. Detecting Fake Account: Now consider the same problem of finding the majority element, an account is called fake account if the images present in the folder does not have any majority element. Improvise in your algorithm for part A to make this algorithm. (pseudo code required)

The Challenge 3: NUTS AND BOLTS (pseudo code required) (15 points)

You are Given two piles of n locks of different sizes and n keys of different sizes, where you don't know which lock has which key. For every lock there is exactly one key. Find out the locks and their corresponding keys efficiently. Note that You cannot do Comparison between locks and keys, the only way to check that is if the lock gets fit by the key then its his key, otherwise we can see if the key is bigger than the lock or smaller than the lock fitting by trying it.

- Design ALGORITHM which SHOULD TAKE EXPECTED O(N LOG N) TIME. ANALYZE YOUR ALGORITHM's COMPLEXITY. TELL ITS BEST AND WORST CASES.

Now design DETERMINISTIC ALGORITHM which should take exactly O(N log N) time. Analyze your algorithm complexity.

Algo 2020 - Homework 5 - PART 2
CHALLENGES (Taken from Microsoft and Google Interviews)

CHALLENGE # 1 Single Number **(Hint - Use hash map, in STL it is named as unordered_map)**

Given a **non-empty** array of integers, every element appears *twice* except for one. Find that single one.
Note: Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

Example 1: **Input:** [2,2,1] **Output:** 1 **Example 2:** **Input:** [4,1,2,1,2] **Output:** 4	**class Solution {** **public:** **int singleNumber(vector<int>& nums) {** **}** **};**

CHALLENGE # 2 Happy Number **(Hint - Use hash map, in STL it is named as unordered_map)**
Write an algorithm to determine if a number n is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers.

Return True if n is a happy number, and False if not. Algorithm must run in linear time (of the number of elements appear in the chain of numbers)

Input: 19 **Output: true** Explanation: $1^2 + 9^2 = 82$ $8^2 + 2^2 = 68$ $6^2 + 8^2 = 100$ $1^2 + 0^2 + 0^2 = 1$ **The chain is 19, 82, 68, 100, 1**	**Input: 7** **Output: true** Explanation: $7^2 = 49$ $4^2+9^2 = 97$ $9^2+7^2 = 130$ $1^2 + 3^2 = 10$ $1^2 + 0^2 = 1$ **The chain is 7, 49, 97, 130, 10, 1**	**Fill In one more example**

CHALLENGE # 3 Largest contiguous Subarray
Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.
Example:

Input: [-2,1,-3,4,-1,2,1,-5,4], Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.

CHALLENGE # 4 Move Zeroes
Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.
Example:
Input: [0,1,0,3,12] Output: [1,3,12,0,0]
Note:

1. **YOUR ALGORITHM MUST RUN IN O(N) TIME.**
2. You must do this in-place without making a copy of the array.

3. Minimize the total number of operations.

CHALLENGE # 5 Best Time to Buy and Sell Stock II

Say you have an array prices for which the i^{th} element is the price of a given stock on day *i*. Design an algorithm to find the maximum profit. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times).

Note: You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again).

Example 1:	**Example 2:**	**Example 3:**
Input: [7,1,5,3,6,4] Output: 7 Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4. Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.	Input: [1,2,3,4,5] Output: 4 Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4. Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are engaging multiple transactions at the same time. You must sell before buying again.	Input: [7,6,4,3,1] Output: 0 Explanation: In this case, no transaction is done, i.e. max profit = 0.

CHALLENGE # 6 Group Anagrams

Given an array of strings, group anagrams together.

Example:
Input: ["eat", "tea", "tan", "ate", "nat", "bat"],
Output:
[
["ate","eat","tea"],
["nat","tan"],
["bat"]
]
Note:

- All inputs will be in lowercase.
- The order of your output does not matter.

CHALLENGE # 7 Counting Elements

Given an integer array arr, count element x such that x + 1 is also in arr. If there're duplicates in arr, count them separately.

Example 1: Input: arr = [1,2,3] Output: 2 Explanation: 1 and 2 are counted cause 2 and 3 are in arr.
Example 2: Input: arr = [1,1,3,3,5,5,7,7] Output: 0 Explanation: No numbers are counted, cause there's no 2, 4, 6, or 8 in arr.
Example 3: Input: arr = [1,3,2,3,5,0] Output: 3 Explanation: 0, 1 and 2 are counted cause 1, 2 and 3 are in arr.
Example 4: Input: arr = [1,1,2,2] Output: 2 Explanation: Two 1s are counted cause 2 is in arr.
Constraints:

- 1 <= arr.length <= 1000
- 0 <= arr[i] <= 1000