Q1 [Search in a rotated sorted array](https://leetcode.com/problems/search-in-rotated-sorted-array/)

class Solution {

public:

int search(vector<int>& nums, int target) {

int s=0, e=nums.size()-1;

while(s<=e){

int mid = s+(e-s)/2;

if(nums[mid]==target)

return mid;

else if(nums[s]<=nums[mid]){

if(nums[s]<=target && target<nums[mid])

e = mid-1;

else

s = mid+1;

}

else{

if(nums[mid]<target && target<=nums[e])

s = mid+1;

else

e = mid-1;

}

}

return -1;};A screenshot of a computer

Description automatically generated

Q2 [Count triplet with sum smaller than a given value](https://practice.geeksforgeeks.org/problems/count-triplets-with-sum-smaller-than-x5549/1)

long long countTriplets(long long arr[], int n, long long sum)

{

// Your code goes here

sort(arr, arr+n);

long long count = 0;

int i, j, k;

for(int i=0; i<n; i++)

{

j = i+1;

k = n-1;

while(j < k)

{

if(arr[i] + arr[j] + arr[k] == sum)

{

k--;

}

else if(arr[i] + arr[j] + arr[k] > sum)

{

k--;

}

else

{

count += k-j;

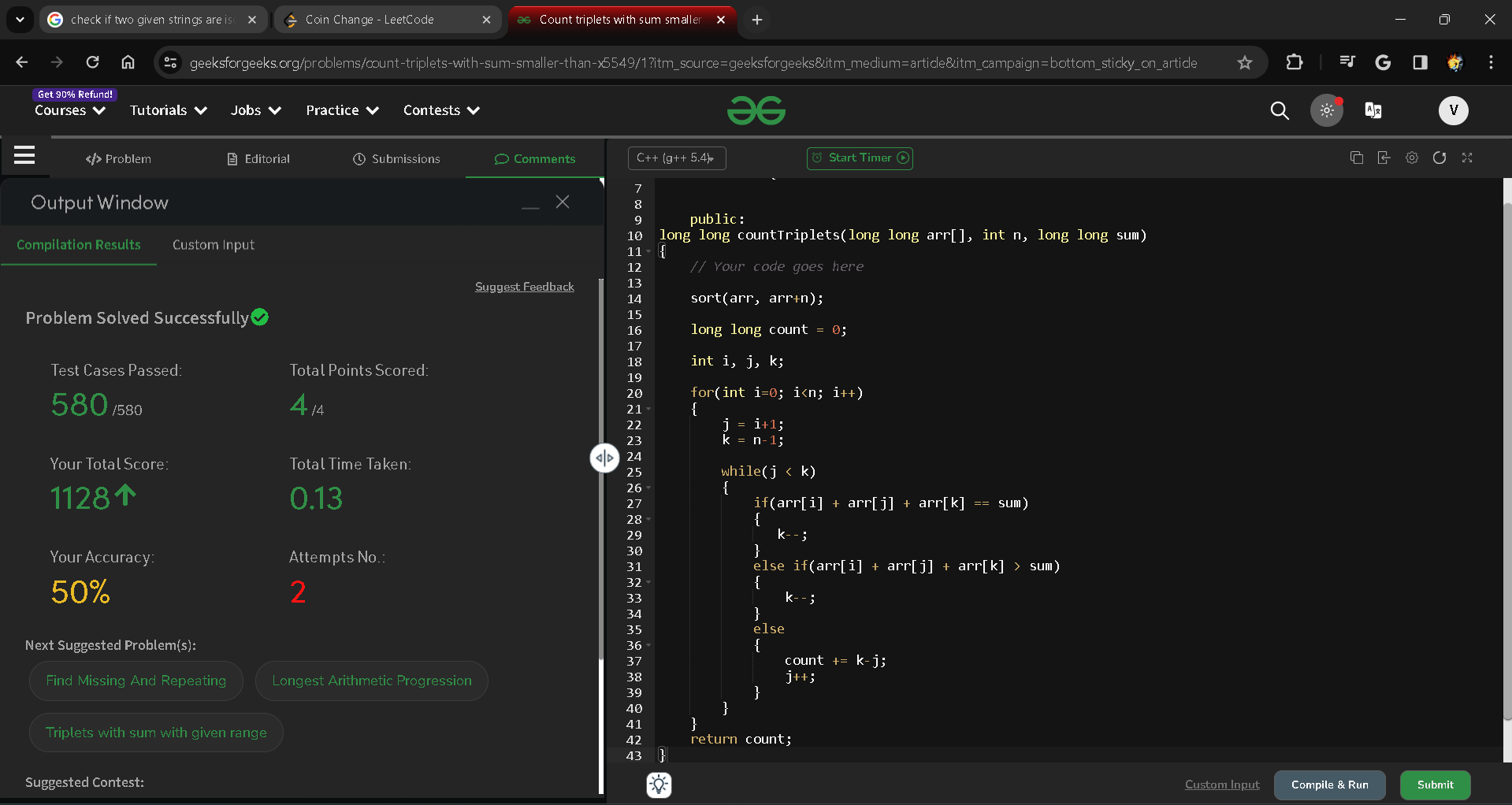
j++;

}

}

}

return count}



Q3 [minimum no. of swaps required to sort the array](https://practice.geeksforgeeks.org/problems/minimum-swaps/1)

int minSwaps(vector<int>&nums)

{

// Code here

int n = nums.size();

vector<pair<int,int>>v;

for(int i = 0; i<n;i++){

v.push\_back({nums[i],i});

}

sort(v.begin(),v.end());

int count =0;

for(int i = 0; i<n;i++){

if(i==v[i].second)

continue;

else{count++;

swap(v[v[i].second],v[i]);

i--;

}

}

return count;

}

A screenshot of a computer program

Description automatically generated

Q4 [K-th Element of Two Sorted Arrays](https://practice.geeksforgeeks.org/problems/k-th-element-of-two-sorted-array/0)

int kthElement(int arr1[], int arr2[], int n, int m, int k)

{

int i=0,j=0,c=0;

while(i<n&&j<m)

{

if(k==0)

return c;

if(arr1[i]<arr2[j])

{

c=arr1[i];

i++;

}

else

{

c=arr2[j];

j++;

}

k--;

}

if(i<n)

return arr1[i+k-1];

return arr2[j+k-1];

}

A screenshot of a computer

Description automatically generated

Q5 [Job Scheduling Algo](https://www.geeksforgeeks.org/weighted-job-scheduling-log-n-time/)

int jobScheduling(vector<int>& startTime, vector<int>& endTime, vector<int>& profit) {

int numJobs = profit.size(); // Number of jobs

vector<tuple<int, int, int>> jobs(numJobs);

for (int i = 0; i < numJobs; ++i) {

jobs[i] = {endTime[i], startTime[i], profit[i]};

}

sort(jobs.begin(), jobs.end());

vector<int> dp(numJobs + 1);

for (int i = 0; i < numJobs; ++i) {

auto [endTime, startTime, profit] = jobs[i];

int latestNonConflictJobIndex = upper\_bound(jobs.begin(), jobs.begin() + i, startTime, [&](int time, const auto& job) -> bool {

return time < get<0>(job);

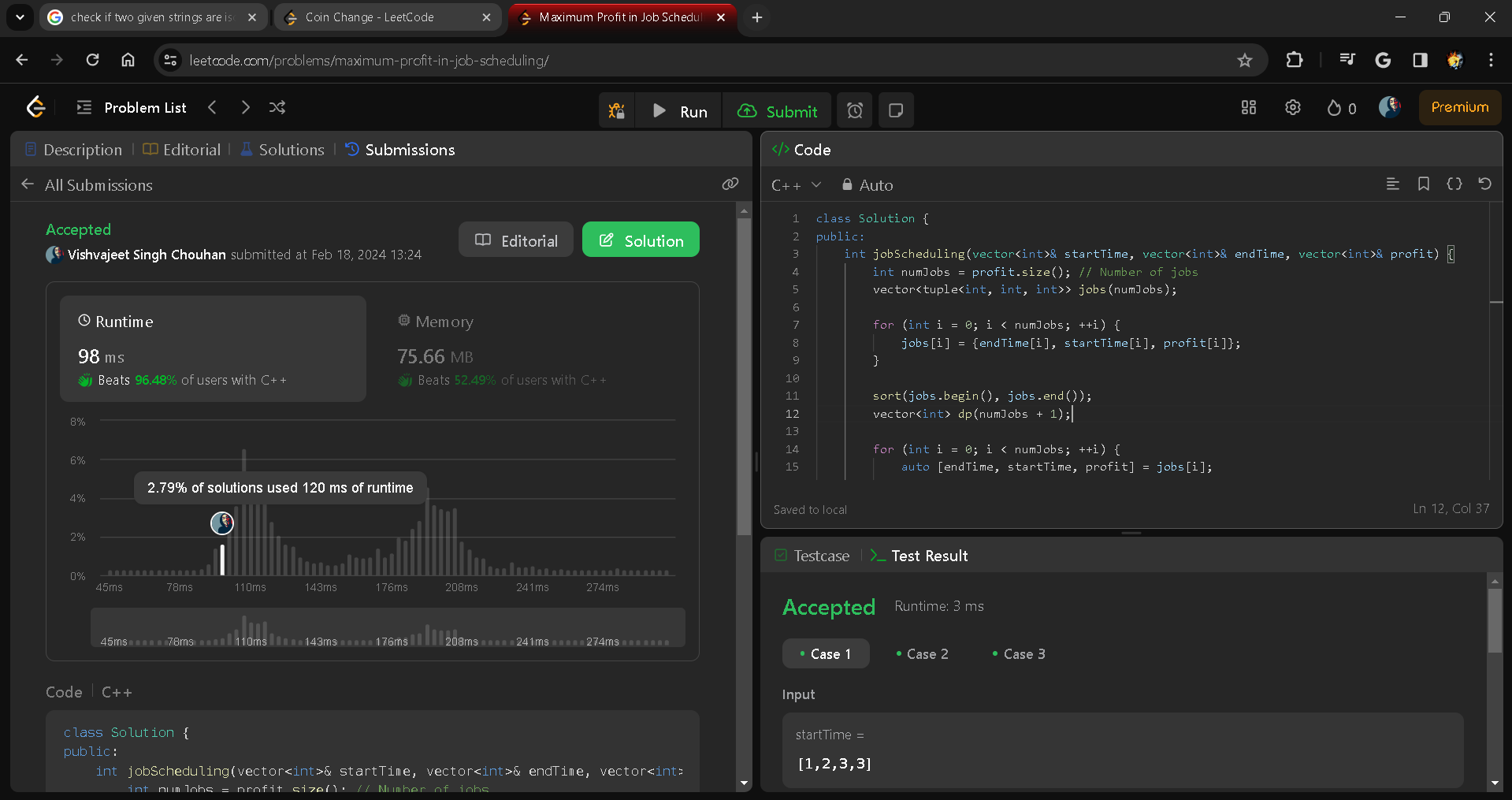
}) - jobs.begin();

dp[i + 1] = max(dp[i], dp[latestNonConflictJobIndex] + profit);

}

return dp[numJobs];

}



Q6 [Subset Sums](https://www.spoj.com/problems/SUBSUMS/)

public static void subsets(int[] arr, int idx, List<List<Integer>>list,List<Integer> ls){

if(idx>=arr.length){

list.add(new ArrayList<>(ls));

return;

}

// don't pick the element

subsets(arr,idx+1,list,ls);

// pick the element

ls.add(arr[idx]);

subsets(arr,idx+1,list,ls);

ls.remove(ls.size()-1);

}

public List<List<Integer>> subsets(int[] nums) {

List<List<Integer>> list=new ArrayList<>();

List<Integer> ls=new ArrayList<>();

subsets(nums,0,list,ls);

return list;

}

