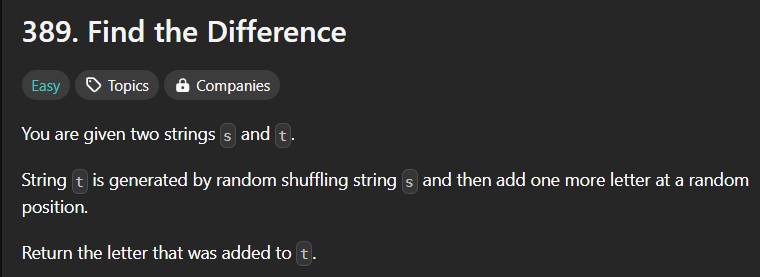
**ASSIGNMENT 5**

**Student Name:** Suryanshu **UID:** 22BCS12106  **Branch:** CSE **Section:** 22BCS\_IOT\_605 B

**Semester:** 6th **DOP:**05-03-2025

**Subject:** Advanced Programming Lab-II **Subject Code:** 22CSP-351 **Question 1**

**Code:**

class Solution { public:

char findTheDifference(string s, string t) { int s\_sum = 0, t\_sum = 0;

for (char c : s) { s\_sum += int(c);

}

for (char c : t) { t\_sum += int(c);

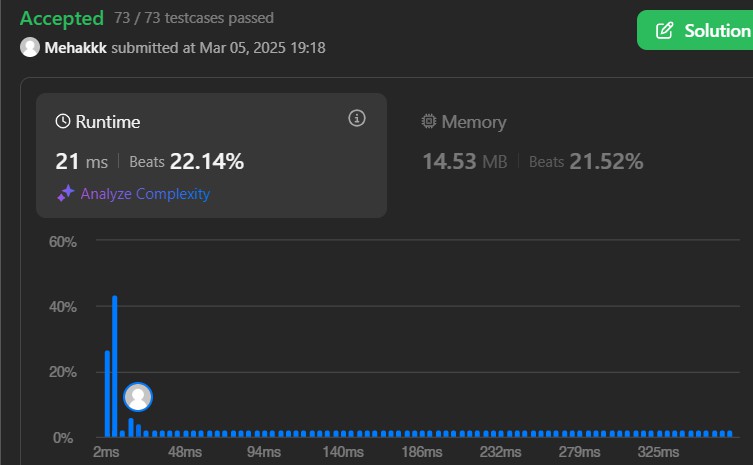
}

return char(t\_sum - s\_sum);

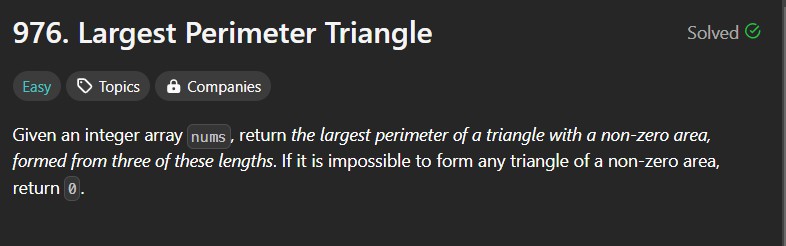
}

};

**Output:**



**Question 2**



**Code:**

class Solution { public:

int largestPerimeter(vector<int>& nums) { sort(nums.begin(),nums.end());

for(int i=nums.size()-1;i>1;i--){

if(nums[i]<nums[i-1]+nums[i-2]){

return nums[i]+nums[i-1]+nums[i-2];

}

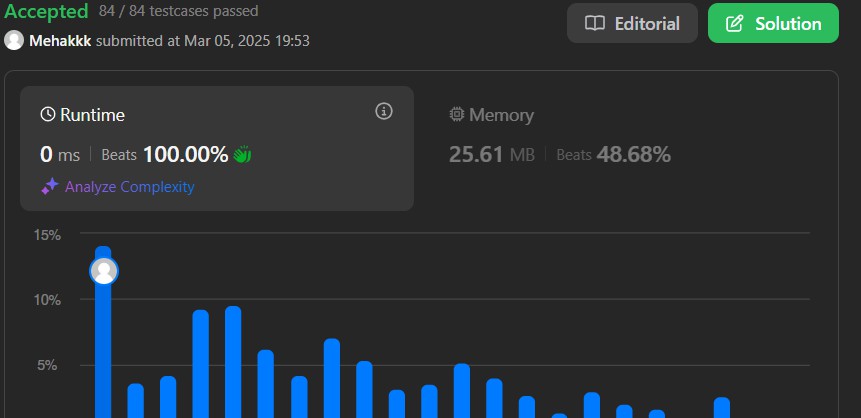
}

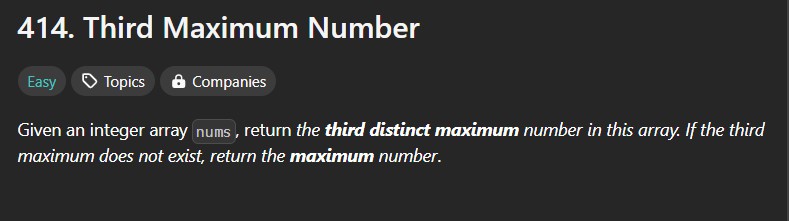
return 0;

}

};

**Output:**



**Question 3**

**Code:**

class Solution { public:

int thirdMax(vector<int>& nums) {

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

int largest,seclargest,thirdlargest;

largest= nums[0];

seclargest=nums[0];

thirdlargest=nums[0];

for(int i=0;i<nums.size();i++){

if(nums[i]>largest){

thirdlargest=seclargest;

seclargest=largest;

largest=nums[i];

}

else if(nums[i]>seclargest && nums[i]<largest){

thirdlargest=seclargest;

seclargest=nums[i];

}

else if(nums[i]>thirdlargest && nums[i]<seclargest){

thirdlargest=nums[i];

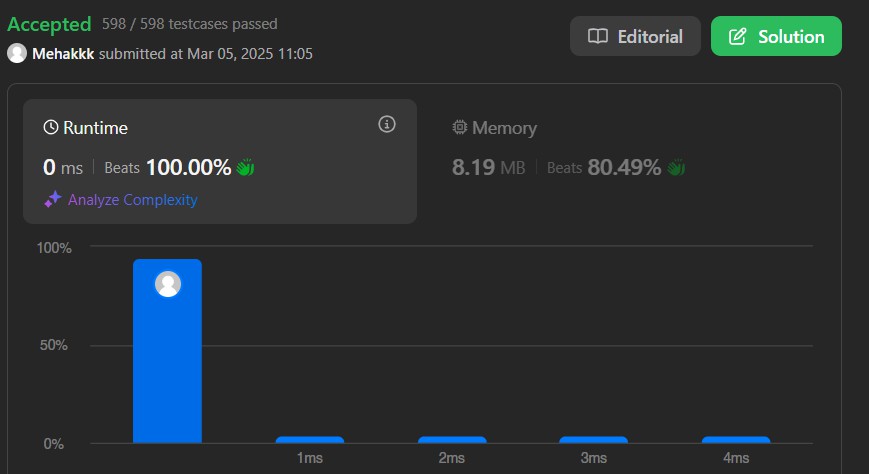
}

}

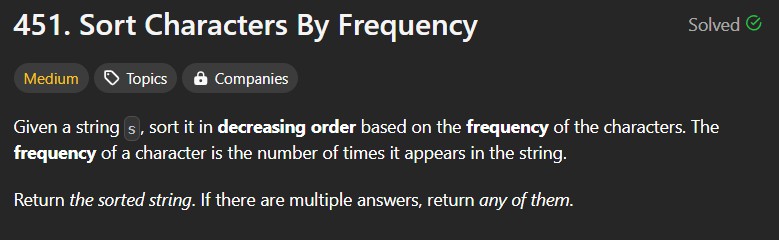
return ((nums.size()<=2 || seclargest==thirdlargest)?largest:thirdlargest);

}

};

**Output:**

**Question 4**



**Code:**

class Solution { public:

string frequencySort(string s) { unordered\_map<char,int> map; vector<pair<int,char>>v; string res;

for(int i=0;i<s.length();i++){ map[s[i]]++;

}

for(auto it: map){ v.push\_back({it.second,it.first});

}

sort(v.rbegin(),v.rend());

for(auto it : v){

for(int j=0;j<it.first;j++){ res+=it.second;

}

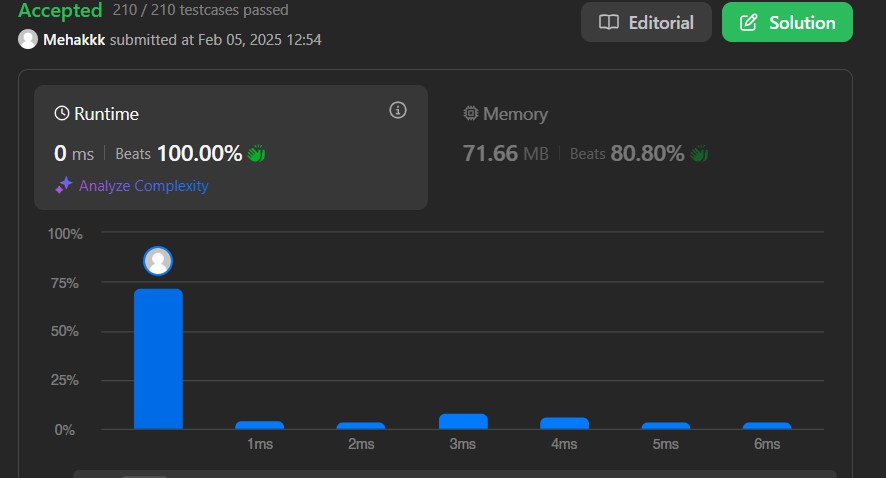
}

return res;

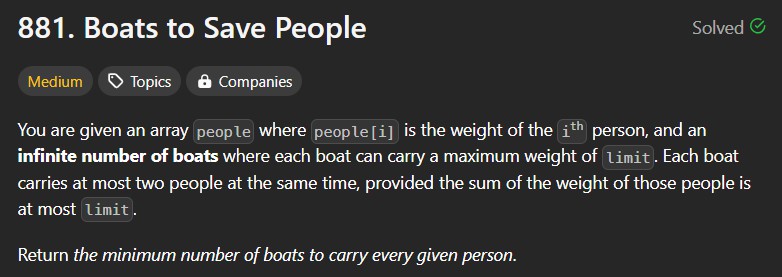
}

};

**Output:**



**Question 5**



**Code:**

class Solution { public:

int numRescueBoats(vector<int>& people, int limit) { sort(people.begin(), people.end());

int i=0, boats=0;

int n= people.size()-1;

while(i<=n){

if(people[i]+people[n]<=limit){ i++;

n--;

boats++;

}

else{

boats++; n--;

}

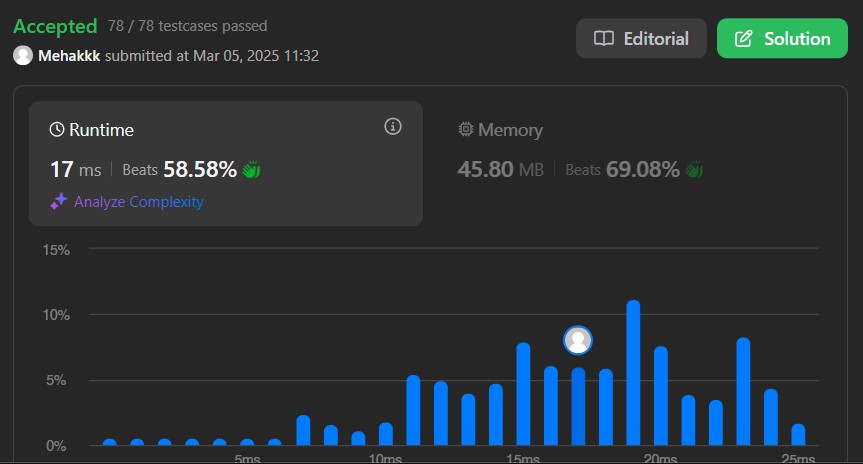
}

return boats;

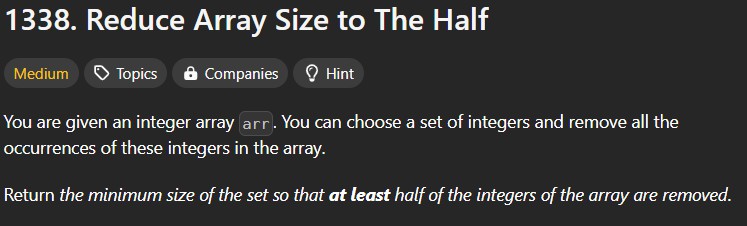
}

};

**Output:**



**Question 6**



**Code**

class Solution { public:

int minSetSize(vector<int>& arr)

{

map<int,int>mp;

for(auto val:arr) mp[val]++;

priority\_queue<int>pq;

for(auto [val, cnt]:mp)

pq.push(cnt);

int ans = 0, need = arr.size()/2;

while(need > 0)

{

ans++;

need -= pq.top(); pq.pop();

}

return ans;

}

};

**Output**

