Assignment -5

**Student Name:** Mayank Vashishta **UID:** 22BCS12920

**Branch:**CSE **Section/Group:**605-B

**Semester:** 5 **Date of Performance:** 05/03/25

**Subject Name:** AP **Subject Code: 22CSP-351**

**Q.1 389. Find the Difference**

class Solution { public:

char findTheDifference(string s, string t) { for(int i=0;i<s.size();i++){

t[i+1]+=t[i] - s[i];

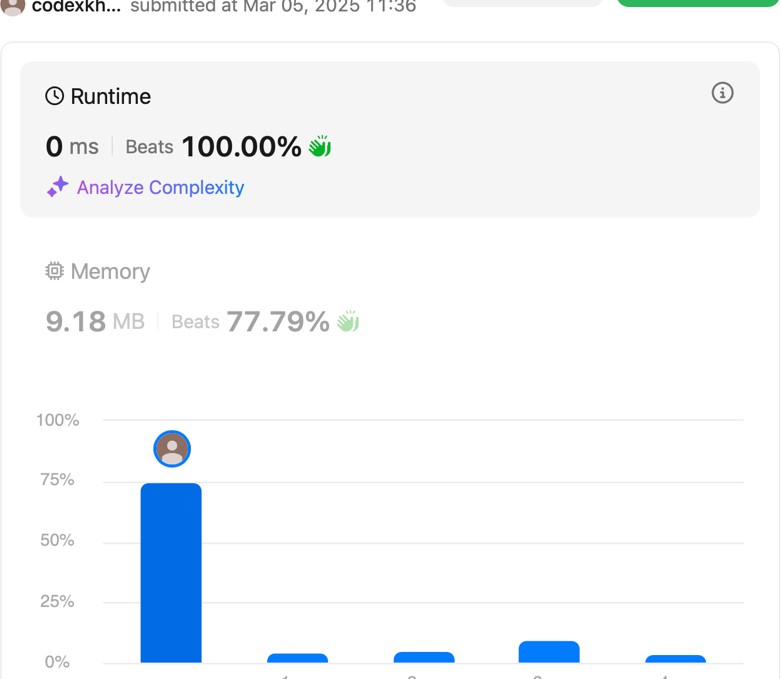
}

return t[t.size()-1];

}

};

Output :



**Q.2**

**414. Third Maximum Number**

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];

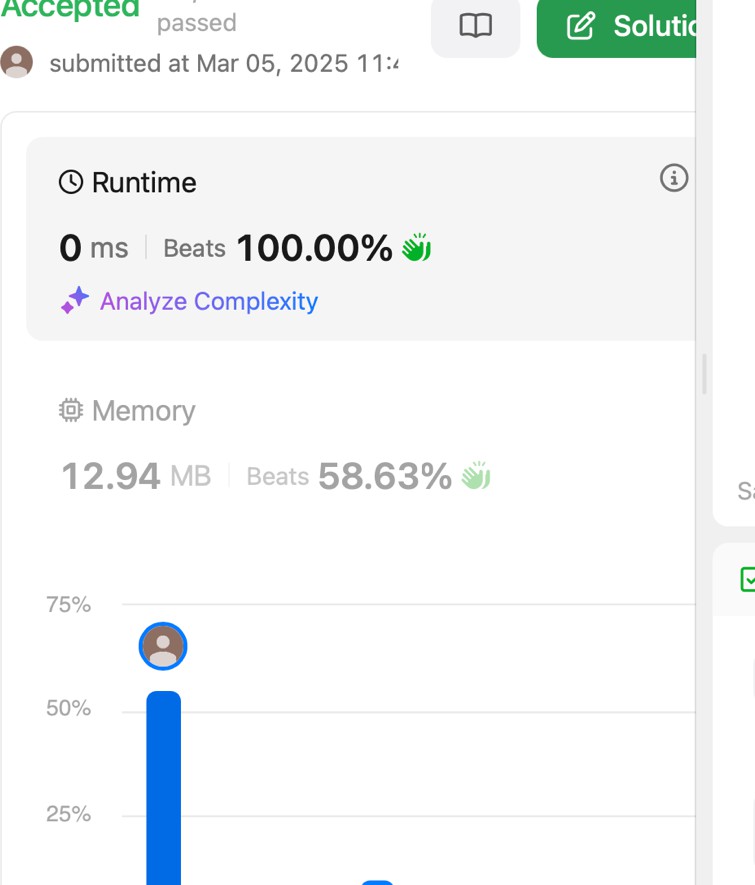
}

}

return 0;

}

};

Output : 

**\**

# Q.3 451. Sort Characters By Frequency

class Solution { public:

typedef pair<char, int> P;

string frequencySort(string s) { vector<P> vec(123);

for(char &ch : s) {

int freq = vec[ch].second; vec[ch] = {ch, freq+1};

}

auto comp = [&](P &p1, P &p2) {

return p1.second > p2.second;

};

sort(begin(vec), end(vec), comp); string result = "";

for(int j = 0; j <= 122; j++) {

result += string(vec[j].second, vec[j].first);

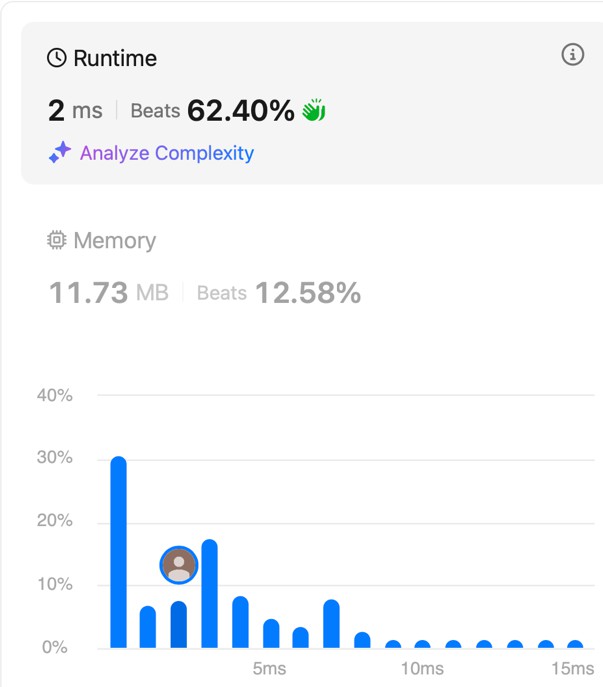
}

return result;

}

};

Output :



# Q.4 452. Minimum Number of Arrows to Burst Balloons

class Solution { public:

int findMinArrowShots(vector<vector<int>>& points) { int n = points.size();

sort(begin(points), end(points));

vector<int> prev = points[0]; int count = 1;

for (int i = 1; i < n; i++) { int currSp = points[i][0]; int currEp = points[i][1];

int prevSp = prev[0]; int prevEp = prev[1];

if (currSp > prevEp) { count++;

prev = points[i];

} else {

prev[0] = max(prevSp, currSp); prev[1] = min(currEp, prevEp);

}

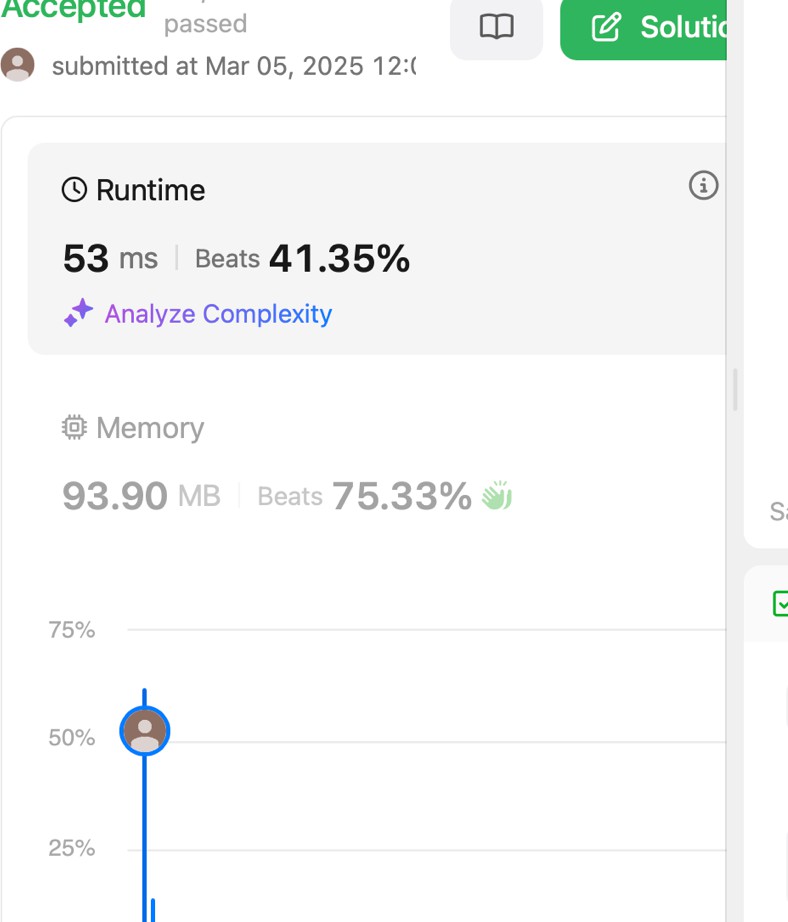
}

return count;

}

};

OUTPUT :



# Q.6. 881. Boats to Save People

class Solution { public:

int numRescueBoats(vector<int>& people, int limit) { int n = people.size();

sort(begin(people) , end(people));

int i = 0 ; int j = n-1;

int boats = 0 ;

while(i <= j){

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

j--;

}

else {

j--;

}

boats++ ;

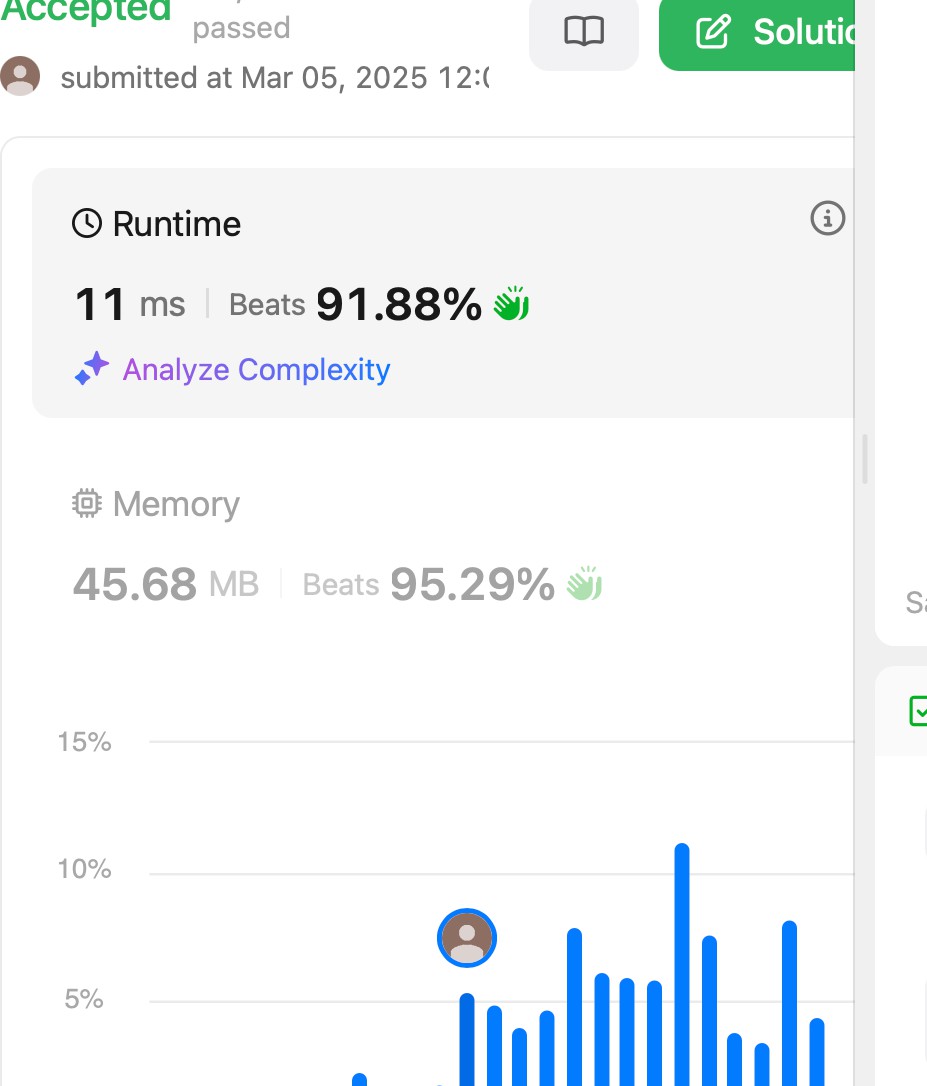
}

return boats;

}

};

Output :



# Q.7. 973. K Closest Points to Origin

#include <queue> #include <vector> using namespace std;

class Solution { public:

vector<vector<int>> kClosest(vector<vector<int>>& points, int k) { priority\_queue<pair<int, vector<int>>> maxHeap;

for (const auto& point : points) { int x = point[0];

int y = point[1];

int distance = x \* x + y \* y;

maxHeap.push({distance, point}); if (maxHeap.size() > k) {

maxHeap.pop();

}

}

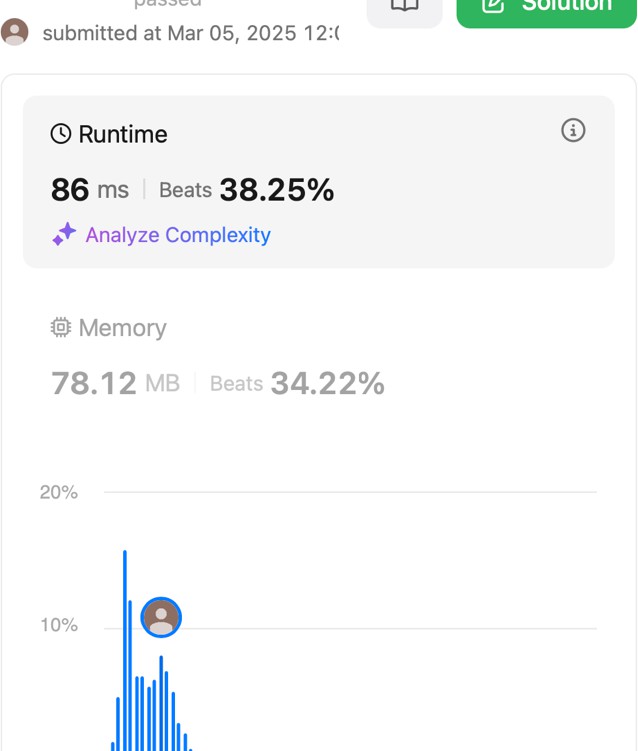
vector<vector<int>> result; while (!maxHeap.empty()) {

result.push\_back(maxHeap.top().second); maxHeap.pop(); }

return result;

}

};

Output:

# Q.8 1338. Reduce Array Size to The Half

class Solution { public:

int minSetSize(vector<int>& arr) { unordered\_map<int, int> cnt; for (int x : arr) ++cnt[x];

vector<int> frequencies;

for (auto [\_, freq] : cnt) frequencies.push\_back(freq); sort(frequencies.begin(), frequencies.end());

int ans = 0, removed = 0, half = arr.size() / 2, i = frequencies.size() - 1; while (removed < half) {

ans += 1;

removed += frequencies[i--];

}

return ans;

}

};

Output :

