

## ASSIGNMENT-5

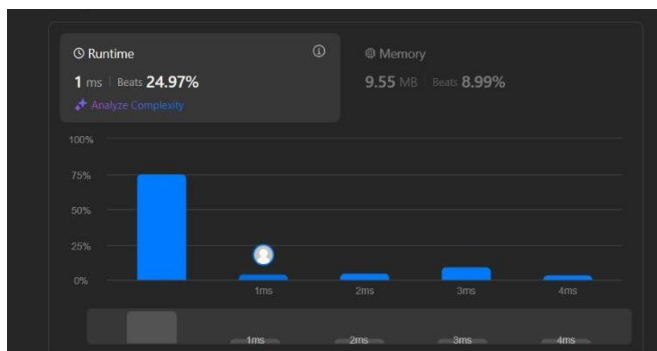
**Name:** Rupesh Bansal

**Section:** FL\_IOT-603/A

**UID:** 22BCS50190

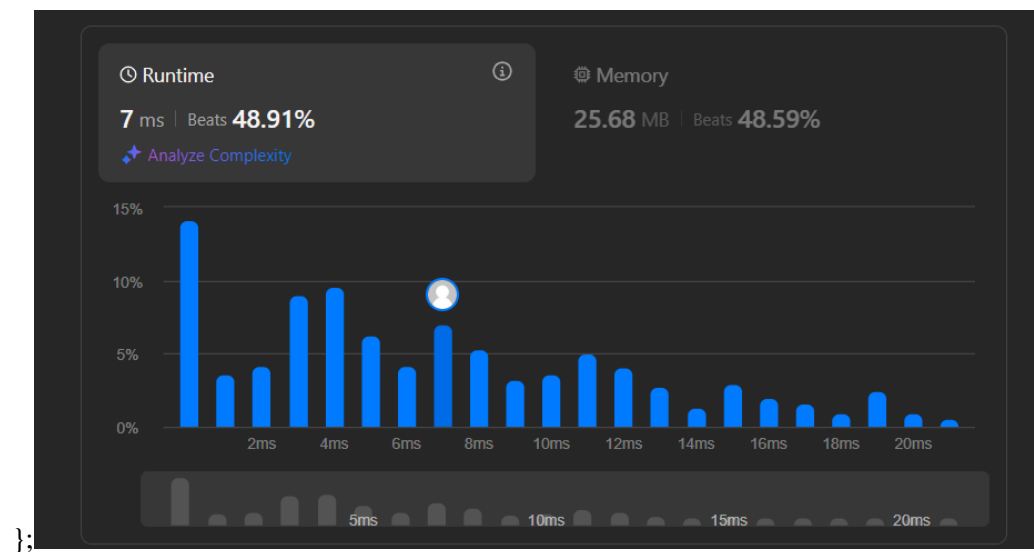
### 389. Find the difference

```
class Solution {  
public:  
    char findTheDifference(string s, string t) {  
        unordered_map<char,int>mpp;  
        for(int i=0;i<t.length();i++){  
            mpp[t[i]]++;  
        }  
        for(int i=0;i<s.length();i++){  
            mpp[s[i]]--;  
        }  
        for(auto it:mpp){  
            if(it.second>0){  
                return it.first;  
            }  
        }  
        return '0';  
    }  
};
```



### 976. Largest Perimeter Triangle

```
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;
    }
};
```



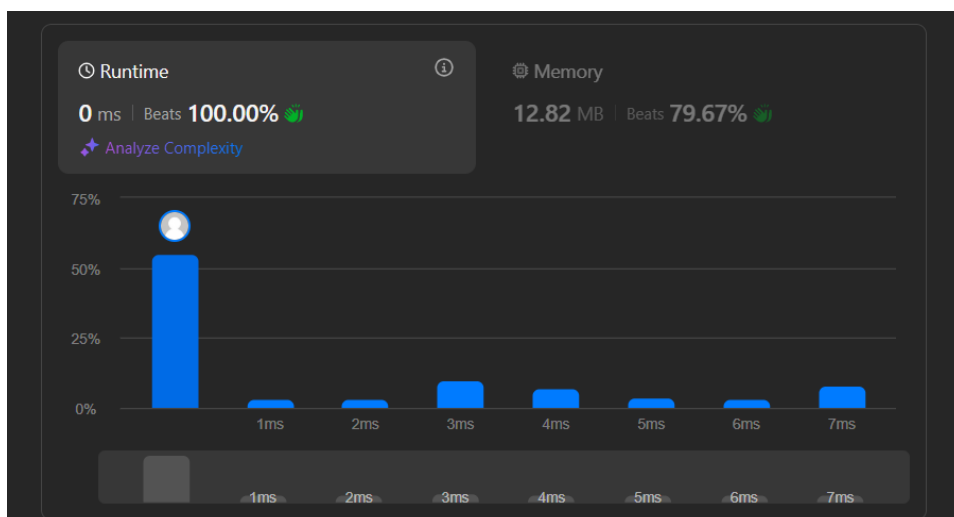
### 414. Third Maximum Number

```
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;
}
};

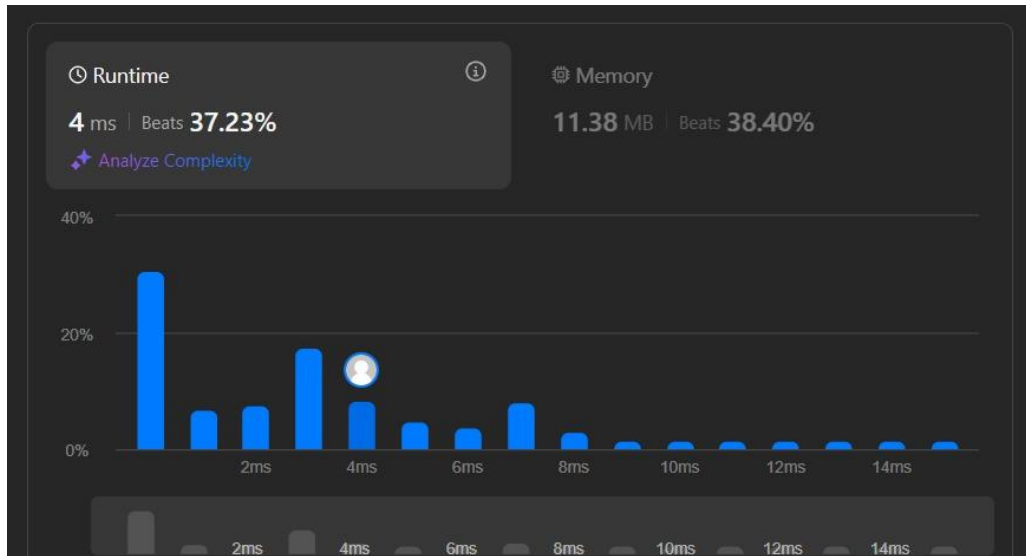
```



#### 451. Sort Characters By Frequency

```
class Solution {
public:
    string frequencySort(string s) {
        auto cmp = [](const pair<char, int>& a, const pair<char, int>& b) {
            return a.second < b.second;
        };
        priority_queue<pair<char, int>, vector<pair<char, int>>, decltype(cmp)> pq(cmp);
        unordered_map<char, int> hm;
        for (char c : s) {
            hm[c]++;
        }
        for (const auto& entry : hm) {
            pq.push(make_pair(entry.first, entry.second));
        }
        string result = "";
        while (!pq.empty()) {
            pair<char, int> p = pq.top();
            pq.pop();
            result.append(p.second, p.first);
        }

        return result;
    }
};
```



### 881. Boats to Save People

```
class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {
        // sort vector
        sort(people.begin(),people.end());
        int i = 0, j = people.size() - 1,cnt = 0;
        while(i <= j)
        {
            // lightest person + heaviest person sum <= limit
            // they can go together
            if(people[i] + people[j] <= limit)
            {
                ++i;
                --j;
            }
            // if sum is over the limit,
            // heaviest will go alone.
            else
                --j;
        }
    }
};
```

```

        ++cnt; // number of boats
    }
    return cnt;
}
};

```

