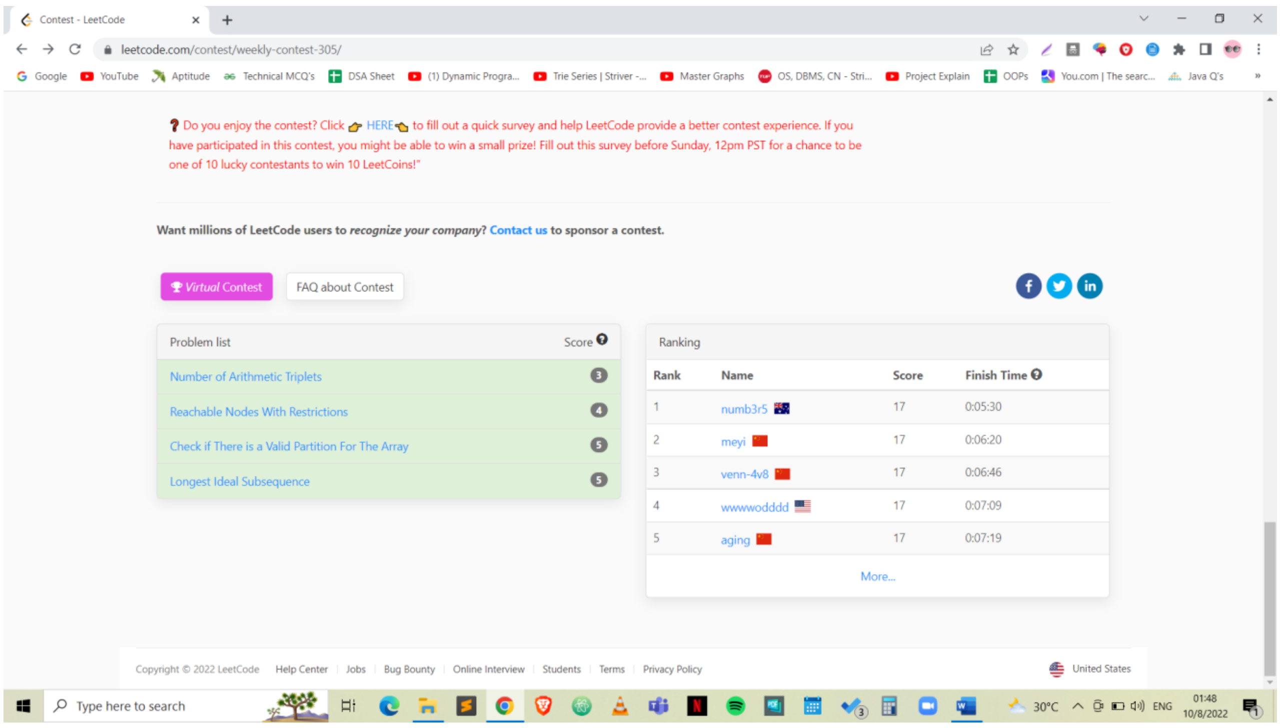
Name: Sagar Srivastava

Roll No: 1900290100127

LeetCode Weekly Contest 305

Week 5

In this contest, I solved 4/4 questions and got a rank of **2795 / 25939**



❓ Do you enjoy the contest? Click 👉 HERE 👈 to fill out a quick survey and help LeetCode provide a better contest experience. If you have participated in this contest, you might be able to win a small prize! Fill out this survey before Sunday, 12pm PST for a chance to be one of 10 lucky contestants to win 10 LeetCoins!"

Want millions of LeetCode users to *recognize your company*? Contact us to sponsor a contest.

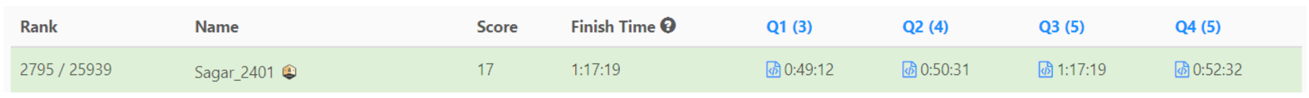
Problem list	Score
Number of Arithmetic Triplets	3
Reachable Nodes With Restrictions	4
Check if There is a Valid Partition For The Array	5
Longest Ideal Subsequence	5

Ranking

Rank	Name	Score	Finish Time
1	numb3r5	17	0:05:30
2	meyi	17	0:06:20
3	venn-4v8	17	0:06:46
4	wwwwodddd	17	0:07:09
5	aging	17	0:07:19

More...

Copyright © 2022 LeetCode Help Center | Jobs | Bug Bounty | Online Interview | Students | Terms | Privacy Policy United States



Rank	Name	Score	Finish Time	Q1 (3)	Q2 (4)	Q3 (5)	Q4 (5)
2795 / 25939	Sagar_2401	17	1:17:19	0:49:12	0:50:31	1:17:19	0:52:32

The questions that I've solved are below:

1) Number of Arithmetic Triplets
2) Reachable Nodes with Restrictions
3) Check if there is a valid partition for the array
4) Longest Ideal Subsequence

1) Number of Arithmetic Triplets

```
class Solution {
public:
    int arithmeticTriplets(vector<int>& nums, int diff) {
        int ans = 0;
        for(int i=0;i<nums.size();i++){
            for(int j=i+1;j<nums.size();j++){
                for(int k=j+1;k<nums.size();k++){
                    if(nums[j]-nums[i] == nums[k]-nums[j] and nums[j]-nums[i] == diff){
                        ans++;
                    }
                }
            }
        }

        return ans;
    }
};
```

2) Reachable Nodes with Restrictions

```
class Solution {
public:
    int reachableNodes(int n, vector<vector<int>>& edges, vector<int>& restricted) {
        vector<int> adj[n];
```

```
for(auto x : edges){
    int a = x[0];
    int b = x[1];

    adj[a].push_back(b);
    adj[b].push_back(a);
}

vector<int> visited(n,false);
int ans = 0;

for(auto x : restricted){
    visited[x] = true;
}

if(!visited[0]){
    queue<int> q;
    q.push(0);
    visited[0] = true;
    while(!q.empty()){
        ans++;
        int node = q.front();
        q.pop();
        for(auto it : adj[node]){
             if(visited[it] == false){
                        q.push(it);
                        visited[it] = true;
             }
        }
    }
}
```

```
        return ans;
    }
};
```

3) Check if there is a valid partition for the array

```
class Solution {
public:

    bool solve(int i, vector<int> &nums, vector<int> &dp){
        if(i == nums.size())    return true;
        if(i > nums.size()) return false;

        if(dp[i] != -1) return dp[i];

        if(i+1 < nums.size() and nums[i] == nums[i+1]){
            if(solve(i+2, nums, dp) == true)
                return true;

            if(i+2 < nums.size() and nums[i] == nums[i+2]){
                if(solve(i+3,nums,dp) == true)
                    return true;
            }
        }

        if(i+2 < nums.size() and nums[i] == nums[i+1] - 1 and nums[i] ==
nums[i+2] - 2){
            if(solve(i+3,nums,dp) == true)
                return true;
        }
        return dp[i] = false;
     }

    bool validPartition(vector<int>& nums) {
        if(nums.size() == 2){
            return nums[0] == nums[1];
        }

        vector<int> dp(nums.size(),-1);
        return solve(0,nums,dp);
    }
};
```

4) Longest Ideal Subsequence

```
class Solution {
public:
    int longestIdealString(string s, int k) {
        int n = s.length();
        int dp[26] = {};

        for(int i=0;i<n;i++){
            int a = s[i]-'a';
            int b = dp[a];

            for(int j=0;j<26;j++){
                if(abs(j-a) <= k){
                    b = max(b,dp[j]+1);
                }
            }
            dp[a] = b;
        }

        int ans = 0;

        for(int i=0;i<26;i++){
            ans = max(ans, dp[i]);
        }

        return ans;
    }
};
```