

# DSA Assignment - 4 July

Name: Aditya Tiwari

Roll No: 23/11/EC/040

LeetCode ID: adityatiwari1305t

GeeksForGeeks ID: aditya1305t

Github Repo Link:

[https://github.com/Aditya1305T/SOE\\_Training\\_25](https://github.com/Aditya1305T/SOE_Training_25)

## Question 1: Find the Celebrity

Platform: GeeksForGeeks

Link: - <https://www.geeksforgeeks.org/problems/the-celebrity-problem/1>

The screenshot shows the GeeksForGeeks interface for the 'Find the Celebrity' problem. The 'Problem Solved Successfully' banner is prominent, indicating 1120/1120 test cases passed with 100% accuracy. The user's score is 4/4, and their total score is 31. The 'Solve Next' section suggests 'Maximum Index', 'Stock span problem', and 'Asteroid Collision'. The 'Suggested Contest' section mentions a 'Job-A-Thon Hiring Challenge'. The code editor on the right shows a C++ solution using a stack to find the celebrity.

```
C++ (12)
11 // ...
12 mp[j]++;
13 if(mp[j] == c){
14     st.push(j);
15 }
16 }
17 if(st.empty()){
18     return -1;
19 }
20 stack<int> s;
21 while(!st.empty()){
22     s.push(st.top());
23     st.pop();
24 }
25 for(int i = 0; i < c; i++){
26     int count = 0;
27     for(int j = 0; j < c; j++){
28         if(mp[j][i] == 0){
29             count++;
30         }
31     }
32     if(i == s.top()){
33         if(count == c-1){
34             return i;
35         }
36         if(count != c-1){
37             s.pop();
38             if(s.empty()){
39                 return -1;
40             }
41             continue;
42         }
43     }
44 }
45 return -1;
46 }
47 }
48 }
```

## Question 2: Trapping Rain Water

Platform: LeetCode

Link: - <https://leetcode.com/problems/trapping-rain-water>

The screenshot shows the LeetCode interface for the 'Trapping Rain Water' problem. The 'Accepted' status is confirmed with 324/324 test cases passed. The runtime is 0 ms, beating 100.00% of solutions, and the memory is 26.12 MB, beating 49.88%. The code editor shows a C++ solution using a two-pointer approach to calculate the trapped water.

```
C++
10 while (left < right) {
11     if (leftMax < rightMax) {
12         left++;
13         leftMax = max(leftMax, height[left]);
14         water += leftMax - height[left];
15     } else {
16         right--;
17         rightMax = max(rightMax, height[right]);
18         water += rightMax - height[right];
19     }
20 }
21 return water;
22 }
23 }
24 }
```

Testcase: Case 1, Case 2. Input: height = [0,1,0,2,1,0,1,3,2,1,2,1].

## Question 3: Design LRU Cache

Platform: LeetCode

Link: - <https://leetcode.com/problems/lru-cache/description/>

The screenshot displays the LeetCode submission interface for the "Design LRU Cache" problem. The submission is marked as "Accepted" with 24/24 testcases passed. The performance metrics show a runtime of 101 ms (Beats 36.09%) and memory usage of 186.67 MB (Beats 18.35%). A performance graph is visible, showing the distribution of runtime across different test cases. The C++ code is shown in the editor, implementing the LRU cache logic. The code includes a class LRUCache with a public method get(key) and a private method put(key, value). The implementation uses a doubly linked list to maintain the order of elements and a hash map to store the key-value pairs.

**Runtime:** 101 ms, Beats 36.09%  
**Memory:** 186.67 MB, Beats 18.35%

**Code:**

```
class LRUCache {
public:
    class node {
    public:
        int key;
        int val;
        node* next;
        node* prev;
    };
    LRUCache(int capacity) {
        m.size() = capacity;
        m.erase(tail->prev->key);
        deletenode(tail->prev);
        addnode(new node(key, value));
        m[key_] = head->next;
    }
    int get(int key) {
        if (m.find(key) != m.end()) {
            int param_1 = obj->get(key);
            obj->put(key, value);
        }
    }
};
```

**Testcase:** Test Result

**Case 1:**

["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get"]

[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]