



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Fast Learners Assignment

Student Name: Ashish Kumar

Branch: CSE

Semester: 6

Subject Name: AP Lab

UID:22bcs11958

Section/Group:614(B)

Date of Performance:17/04/25

Subject Code: 22CSP-351

Q1:-Given an $m \times n$ matrix, if an element is 0, set its entire row and column to 0.

```
class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {
        unordered_map<int,int> r1,c1;

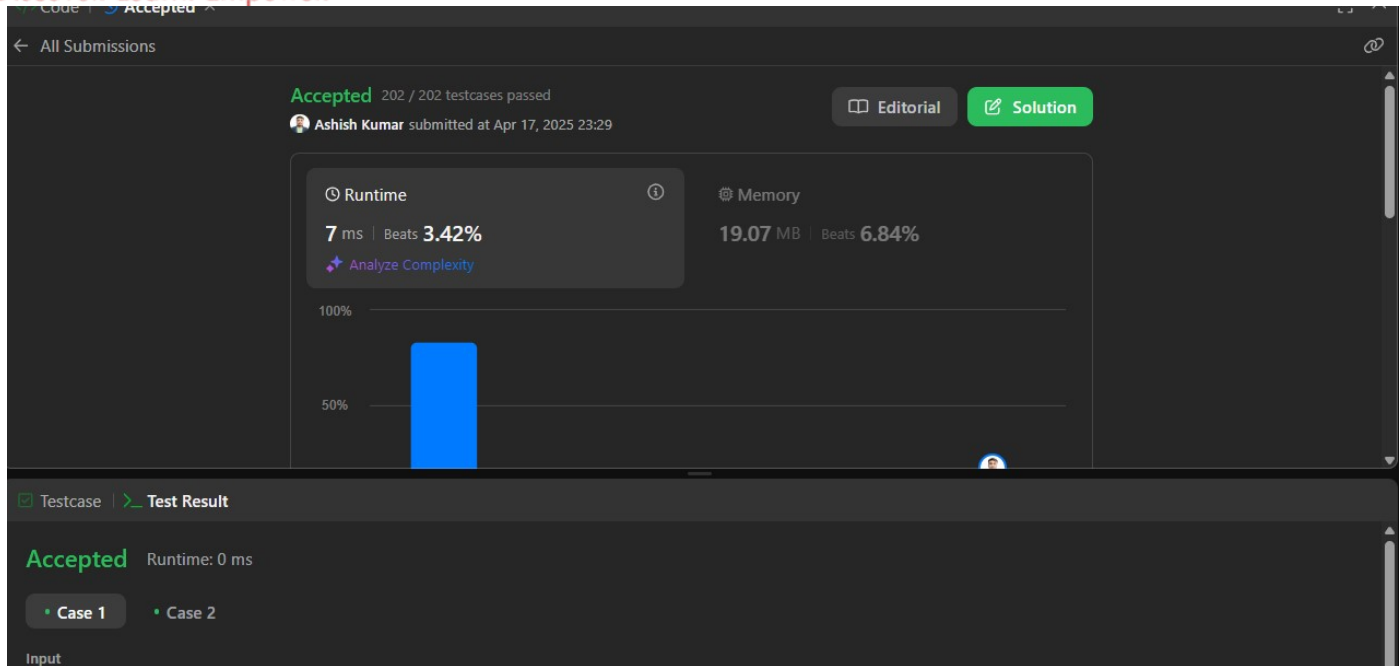
        for(int i = 0; i < matrix.size(); i++){
            for(int j = 0; j < matrix[i].size(); j++){
                if(matrix[i][j] == 0){
                    r1[i] = 1;
                    c1[j] = 1;
                }
            }
        }

        for(int i = 0; i < matrix.size(); i++){
            for(int j = 0; j < matrix[i].size(); j++){
                if(r1[i] == 1 || c1[j] == 1) matrix[i][j] = 0;
            }
        }
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.



Q2:-Given the head of a singly linked list and two integers left and right, reverse the nodes of the list from position left to right

```
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (!head || left == right) return head;

        // Dummy node to simplify edge cases (like reversing from head)
        ListNode* dummy = new ListNode(0);
        dummy->next = head;

        // Step 1: Move `prev` to node before the `left` position
        ListNode* prev = dummy;
        for (int i = 1; i < left; ++i) {
            prev = prev->next;
        }

        // Step 2: Reverse the sublist
        ListNode* curr = prev->next;
        ListNode* next = nullptr;
        for (int i = 0; i < right - left; ++i) {
            next = curr->next;
            curr->next = next->next;
            next->next = prev->next;
            prev->next = next;
        }

        return dummy->next;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

The screenshot shows a submission page for a coding problem. At the top, it says "Accepted" with "44 / 44 testcases passed". The user "Ashish Kumar" submitted at "Apr 17, 2025 23:54". There are buttons for "Editorial" and "Solution". Below this, a box displays performance metrics: "Runtime: 0 ms | Beats 100.00%" and "Memory: 11.17 MB | Beats 72.69%". A link "Analyze Complexity" is also present. A bar chart shows the user's performance relative to others, with a blue bar reaching 100% on the runtime scale. At the bottom, there's a "Testcase" tab and a "Test Result" tab. The "Test Result" tab shows "Accepted" with "Runtime: 0 ms". Below this, there are tabs for "Case 1" and "Case 2". The "Input" section is partially visible at the bottom.

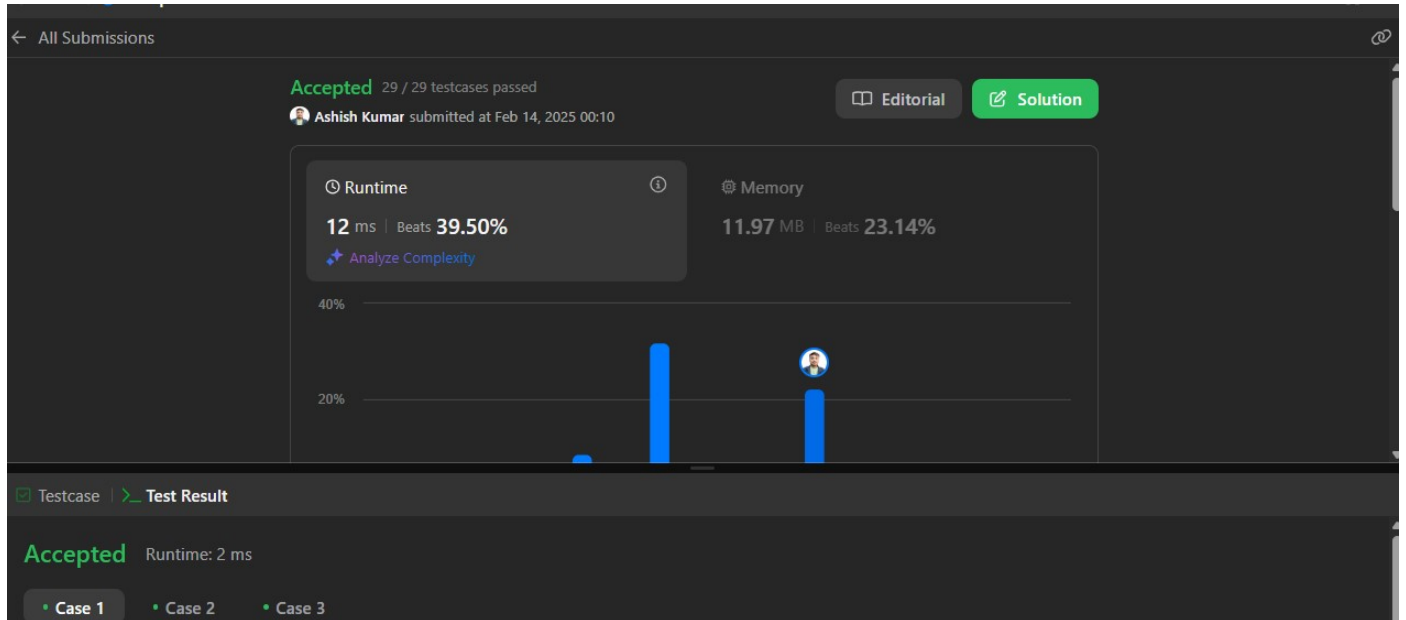
Q3:-Given the head of a linked list, determine whether the linked list contains a cycle. A cycle occurs if a node's next pointer points to a previous node in the list.

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if(head==NULL)
        {
            return false;
        }
        ListNode*fast=head;
        ListNode*slow=head;
        while(fast!=NULL && fast->next!=NULL)
        {
            fast=fast->next->next;
            slow=slow->next;
            if(fast==slow)
            {
                return true;
            }
        }
    }
};
```

```

    }
}
return false;
}
};

```

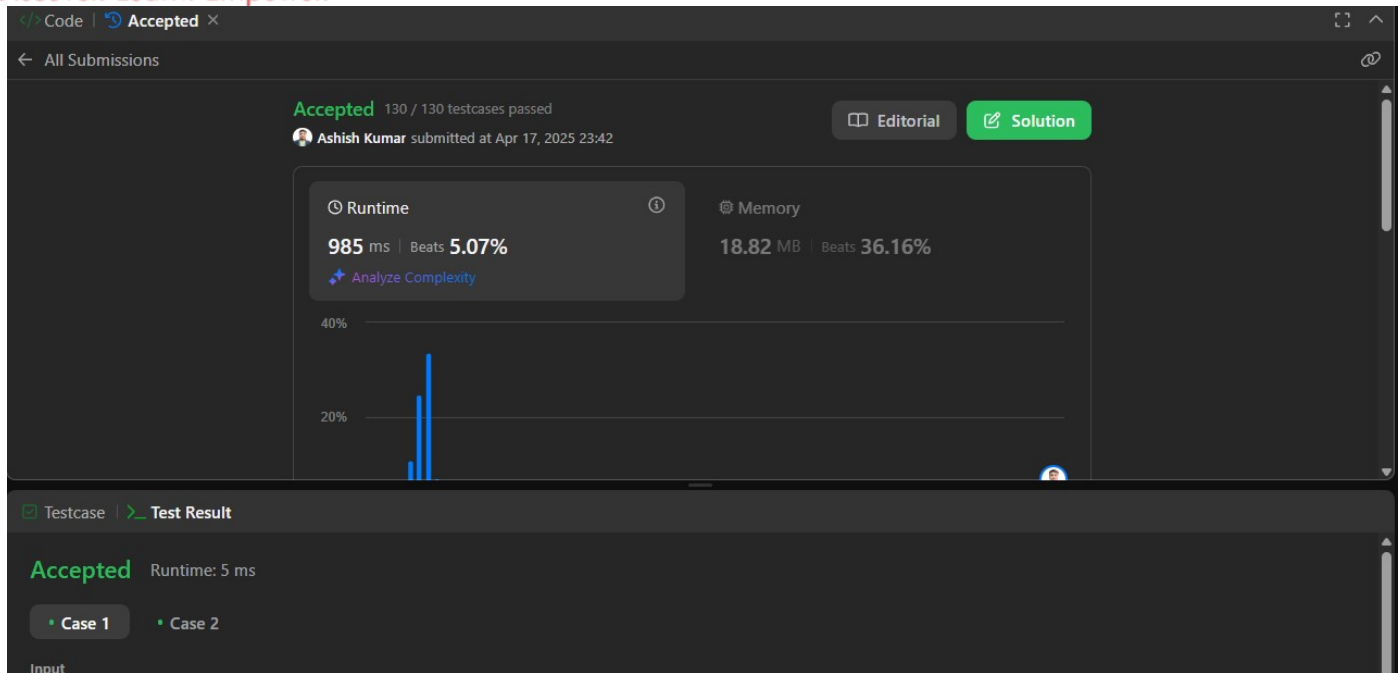


Q4:-Given an $m \times n$ matrix where each row is sorted in ascending order from left to right and each column is sorted in ascending order from top to bottom, and an integer target, determine if the target exists in the matrix.

```

class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        for(int i=0;i<matrix.size();i++)
        {
            for(int j=0;j<matrix[0].size();j++)
            {
                if(matrix[i][j]==target) return true;
            }
        }
        return false;
    }
};

```



Q5:- Given n non-negative integers representing an elevation map where the width of each bar is 1, compute the total amount of water that can be trapped after raining.

```
class Solution {
public:
    int trap(vector<int>& height) {
        int n=height.size();
        int sum=0;
        vector<int> leftMaxheight(n), rightMaxheight(n);
        if(n<=2) return 0;
        leftMaxheight[0]=height[0];
        for(int i=1; i<n; i++)
        {
            leftMaxheight[i]=max(leftMaxheight[i-1], height[i]);
        }
        rightMaxheight[n-1]=height[n-1];
        for(int i=n-2; i>=0; i--)
        {
            rightMaxheight[i]=max(rightMaxheight[i+1], height[i]);
        }
        for(int i=0; i<n; i++)
        {
            sum+=max(0, min(leftMaxheight[i], rightMaxheight[i]) - height[i]);
        }
        return sum;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Code | Accepted

All Submissions

Accepted 323 / 323 testcases passed

Ashish Kumar submitted at Nov 14, 2024 21:52

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

25.98 MB | Beats 80.02%

100%

75%

50%

25%

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2