



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Assignment

Student Name: Mohit Behal

UID: 22BCS11081

Branch: BE-CSE

Section/Group: 22BCS_IOT-614 B

Semester: 06

Date of Performance: 07/04/2025

Subject: Advanced Programming

Subject code: 22CSP-351

- 1. Problem: Set Matrix Zeroes:** Given an $m \times n$ matrix, if an element is 0, set its entire row and column to 0.

Code:

```
class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {
        int m = matrix.size();
        int n = matrix[0].size();

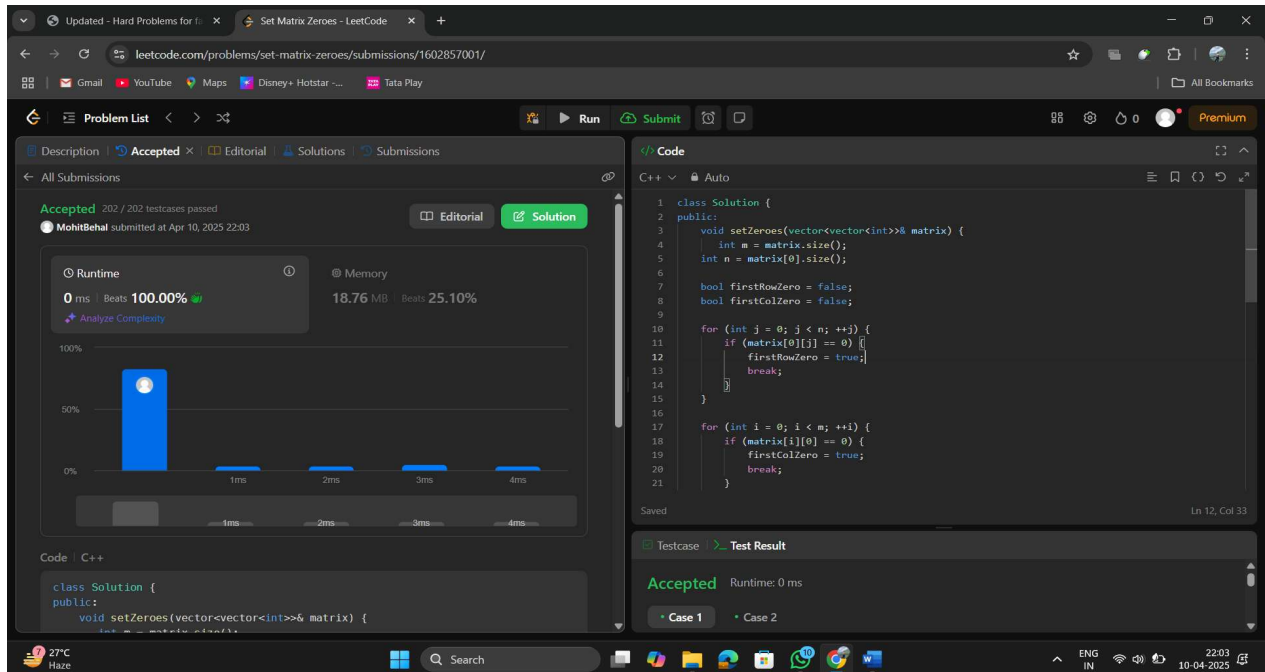
        bool firstRowZero = false;
        bool firstColZero = false;
        for (int j = 0; j < n; ++j) {
            if (matrix[0][j] == 0) {
                firstRowZero = true;
                break;
            }
        }
        for (int i = 0; i < m; ++i) {
            if (matrix[i][0] == 0) {
                firstColZero = true;
                break;
            }
        }
        for (int i = 1; i < m; ++i) {
            for (int j = 1; j < n; ++j) {
                if (matrix[i][j] == 0) {
                    matrix[i][0] = 0;
                    matrix[0][j] = 0;
                }
            }
        }
    }
};
```

```

    }
}
for (int i = 1; i < m; ++i) {
    for (int j = 1; j < n; ++j) {
        if (matrix[i][0] == 0 || matrix[0][j] == 0) {
            matrix[i][j] = 0;
        }
    }
}
if (firstRowZero) {
    for (int j = 0; j < n; ++j) {
        matrix[0][j] = 0;
    }
}
if (firstColZero) {
    for (int i = 0; i < m; ++i) {
        matrix[i][0] = 0;
    }
}
}
};

```

Output:



The screenshot shows a web browser displaying a LeetCode submission for the problem "Set Matrix Zeroes". The submission is marked as "Accepted" with a runtime of 0 ms and memory usage of 18.76 MB. The code is in C++ and implements the solution using first row and column markers.

Code:

```

1 class Solution {
2 public:
3     void setZeroes(vector<vector<int>>& matrix) {
4         int m = matrix.size();
5         int n = matrix[0].size();
6
7         bool firstRowZero = false;
8         bool firstColZero = false;
9
10        for (int j = 0; j < n; ++j) {
11            if (matrix[0][j] == 0) {
12                firstRowZero = true;
13                break;
14            }
15        }
16
17        for (int i = 0; i < m; ++i) {
18            if (matrix[i][0] == 0) {
19                firstColZero = true;
20                break;
21            }
22        }
23
24        for (int i = 0; i < m; ++i) {
25            for (int j = 0; j < n; ++j) {
26                if (firstRowZero || firstColZero) {
27                    matrix[i][j] = 0;
28                }
29            }
30        }
31    }
32 }

```

Testcase: Accepted Runtime: 0 ms

Case 1 **Case 2**

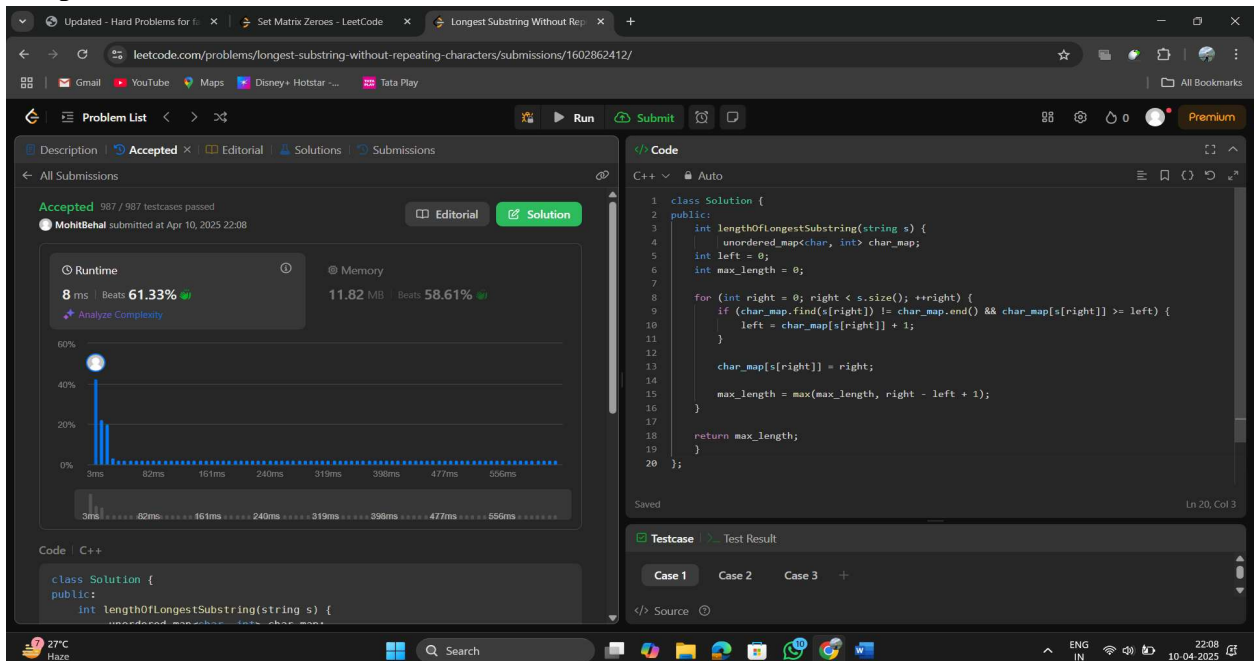
- 2. Problem: Longest Substring Without Repeating Characters:** Given a string *s*, find the length of the longest substring that does not contain any repeating characters.

Code:

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        unordered_map<char, int> char_map;
        int left = 0;
        int max_length = 0;

        for (int right = 0; right < s.size(); ++right) {
            if (char_map.find(s[right]) != char_map.end() && char_map[s[right]] >= left) {
                left = char_map[s[right]] + 1;
            }
            char_map[s[right]] = right;
            max_length = max(max_length, right - left + 1);
        }
        return max_length;
    }
};
```

Output:



- 3. Problem: Reverse Linked List II:** 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.

Code:

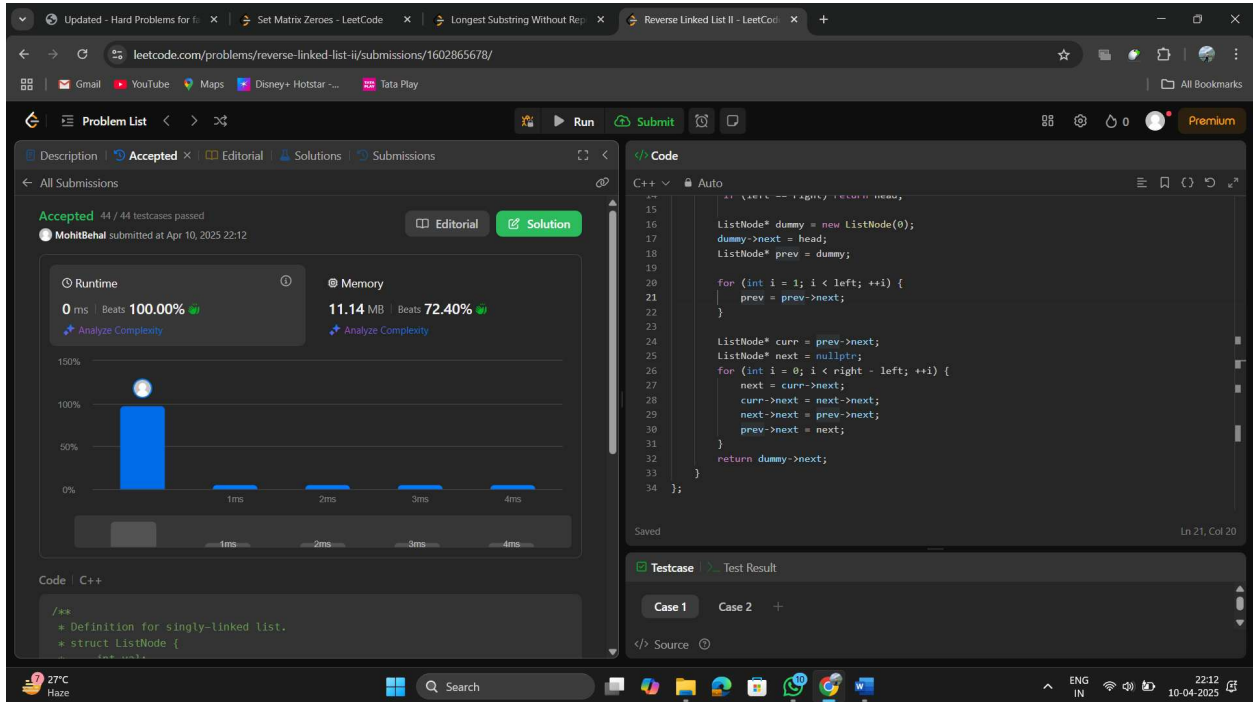
```
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (left == right) return head;

        ListNode* dummy = new ListNode(0);
        dummy->next = head;
        ListNode* prev = dummy;

        for (int i = 1; i < left; ++i) {
            prev = prev->next;
        }

        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;
    }
};
```

Output:



- Problem: Detect a Cycle in a Linked List:** 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.

Code:

```

class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next) return false;

        ListNode *slow = head, *fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast) {
                return true;
            }
        }
    }
};

```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }  
    return false;  
}  
};
```

Output:

