Fast Learner Assignment

Student Name: Dhruv Soni **UID:** 22BCS16772

Branch: BE-CSE **Section/Group:** 22BCS_NTPP-602-A

Semester: 6th **Date of Performance:** 09/04/2025

Subject Name: AP LAB - II **Subject Code:** 22CSP-351

1. **Aim:** Given an m x n integer matrix, if an element is 0, set its entire row and column to 0.

Source Code:

```
class Solution
{ public: void
setZeroes(vector
<vector>&matrix)
{
int m =
matrix.size(), n =
matrix[0].size();
bool
firstRowHasZero
= false,
firstColHasZero =
false; for (int i =
0; i < m; i++) { if
(matrix[i][0] == 0)
firstColHasZero =
true; } for (int j =
0; j < n; j++) { if
(matrix[0][j] == 0)
firstRowHasZero
= true; } 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
(firstColHasZero)
{ for (int i = 0; i <
m; i++) matrix[i]
[0] = 0; } if
(firstRowHasZero
) { for (int j = 0; j
< n; j++) matrix[0]
[j] = 0; } } };
```

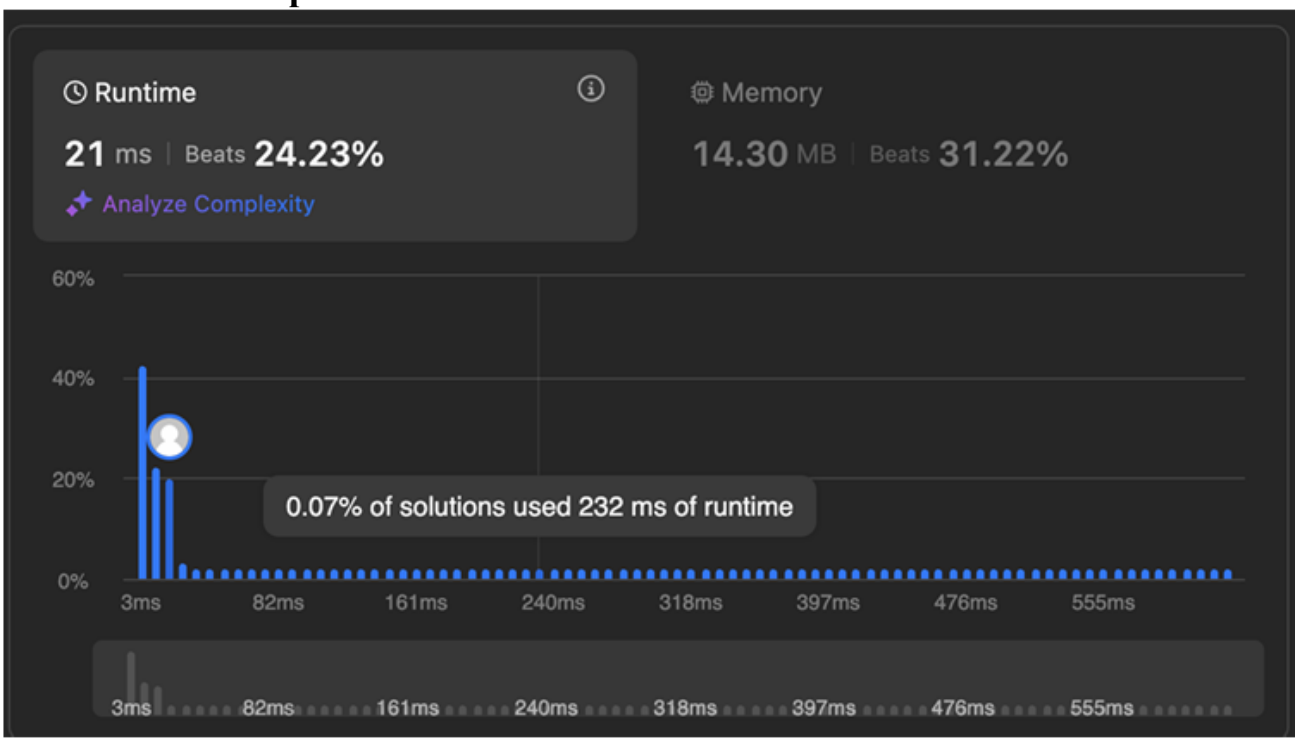
2. **Screenshots of outputs:**

2. Aim: Given a string s, find the length of the longest substring without duplicate characters.

Source Code:

```
class Solution
{
public: int lengthOfLongestSubstring(string s)
{
unordered_set seen;
 int left = 0, maxLength = 0;
 for (int right = 0; right < s.length(); right++)
{
while (seen.find(s[right]) != seen.end())
 {
seen.erase(s[left]); left++; }
seen.insert(s[right]);
maxLength = max(maxLength, right - left + 1);
}
return maxLength; } };
```

Screenshots of outputs:



Aim: Given n non-negative integers representing an elevation map, compute how much water it can trap after raining.

Source Code:

```
class Solution {
public: int trap(vector& height)
{

 if (height.empty()) return 0; int left = 0, right = height.size() - 1;
 int leftMax = 0, rightMax = 0;
int water = 0; while (left < right)
{ if (height[left] < height[right])
{ if (height[left] >= leftMax)
{ leftMax = height[left]; }
 else { water += leftMax - height[left]; }
 left++; } else {
 if (height[right] >= rightMax) { rightMax = height[right]; }
else {
water += rightMax - height[right];
 }
right--;
 } }
return water; } };
```

3. **Screenshots of Outputs:**