

Advanced Programming

ASSIGNMENT 09

Q1. Number of Islands.

Code:

```
C++  Auto
1  #include<vector>
2  using namespace std;
3  class Solution{
4  public:
5  void dfs(vector<vector<char>>&grid,int i,int j){
6  int m=grid.size(),n=grid[0].size();
7  if(i<0||j<0||i>=m||j>=n||grid[i][j]=='0')return;
8  grid[i][j]='0';
9  dfs(grid,i+1,j);
10 dfs(grid,i-1,j);
11 dfs(grid,i,j+1);
12 dfs(grid,i,j-1);
13 }
14 int numIslands(vector<vector<char>>&grid){
15 int m=grid.size(),n=grid[0].size(),count=0;
16 for(int i=0;i<m;++i){
17 for(int j=0;j<n;++j){
18 if(grid[i][j]=='1'){
19 ++count;
20 dfs(grid,i,j);
21 }
22 }
23 }
24 return count;
25 }
26 };
```

Output:

← All Submissions

Accepted 49 / 49 testcases passed

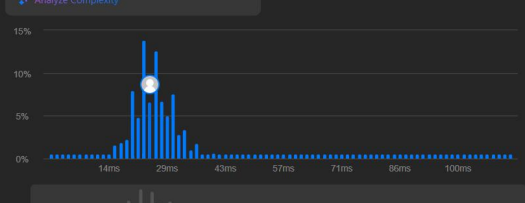
saloni_pundir_ submitted at Apr 16, 2025 12:02

Editorial Solution

Runtime 24 ms | Beats 78.82%

Memory 15.94 MB | Beats 99.59%

Analyze Complexity



14ms 29ms 43ms 57ms 71ms 86ms 100ms

Accepted Runtime: 3 ms

Case 1 Case 2

Input

```
grid =
[[["1","1","1","1","0"],
["1","1","0","1","0"],
["1","1","0","0","0"],
["0","0","0","0","0"]]
```

Output

```
1
```

Expected

```
1
```

Contribute a testcase

Q2. Surrounded Regions.

Code:

```
</> Code | ☒ Testcase | >_ Test Result

C++ ☐ Auto

1 class Solution {
2 public:
3     void solve(vector<vector<char>>& board) {
4         if (board.empty() || board[0].empty()) return;
5         int m = board.size(), n = board[0].size();
6         function<void(int, int)> dfs = [&](int i, int j) {
7             if (i < 0 || i >= m || j < 0 || j >= n || board[i][j] != 'O') return;
8             board[i][j] = 'E';
9             dfs(i - 1, j);
10            dfs(i + 1, j);
11            dfs(i, j - 1);
12            dfs(i, j + 1);
13        };
14
15        for (int i = 0; i < m; i++) {
16            dfs(i, 0);
17            dfs(i, n - 1);
18        }
19        for (int j = 0; j < n; j++) {
20            dfs(0, j);
21            dfs(m - 1, j);
22        }
23        for (int i = 0; i < m; i++) {
24            for (int j = 0; j < n; j++) {
25                if (board[i][j] == 'O') board[i][j] = 'X';
26                if (board[i][j] == 'E') board[i][j] = 'O';
27            }
28        }
29    }
```

Output:

← All Submissions

Accepted 58 / 58 testcases passed

saloni_pundir_ submitted at Apr 16, 2025 12:04

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

15.12 MB | Beats 24.08%

0.06% of solutions used 18 ms of runtime

Code | C++

```
class Solution {
public:
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

board =

```
[["X","X","X","X"],["X","O","O","X"],["X","X","O","X"],["X","O","X","X"]]
```

Output

```
[["X","X","X","X"],["X","X","X","X"],["X","X","X","X"],["X","O","X","X"]]
```

Expected

```
[["X","X","X","X"],["X","X","X","X"],["X","X","X","X"],["X","O","X","X"]]
```

Contribute a testcase

Q3. Friend Circles.

Code:

```
C++ v Auto
1 class Solution {
2 public:
3     int findCircleNum(vector<vector<int>>& isConnected) {
4         int n = isConnected.size();
5         vector<bool> visited(n, false);
6         int provinceCount = 0;
7
8         for (int i = 0; i < n; i++) {
9             if (!visited[i]) {
10                 vector<int> queue;
11                 queue.push_back(i);
12                 visited[i] = true;
13                 bfs1(isConnected, queue, visited);
14                 provinceCount++;
15             }
16         }
17
18         return provinceCount;
19     }
20
21 private:
22     void bfs1(vector<vector<int>>& isConnected, vector<int>& queue, vector<bool>& visited) {
23         if (queue.empty()) return;
24
25         int city = queue.front();
26         queue.erase(queue.begin());
27
28         vector<int> nextQueue;
29         for (int neighbor = 0; neighbor < isConnected.size(); neighbor++) {
```

Output:

Accepted 114 / 114 testcases passed

saloni_pundir_ submitted at Apr 16, 2025 12:06

Editorial Solution

Runtime: 0 ms | Beats: 100.00%
Memory: 20.31 MB | Beats: 42.11%

Analyze Complexity

0.56% of solutions used 12 ms of runtime

Accepted Runtime: 0 ms

Case 1 Case 2

Input

isConnected =
[[1,1,0],[1,1,0],[0,0,1]]

Output

2

Expected

2

Contribute a testcase

Code | C++

```
class Solution {
```

Q4. Lowest Common Ancestor of a Binary Tree.

Code:

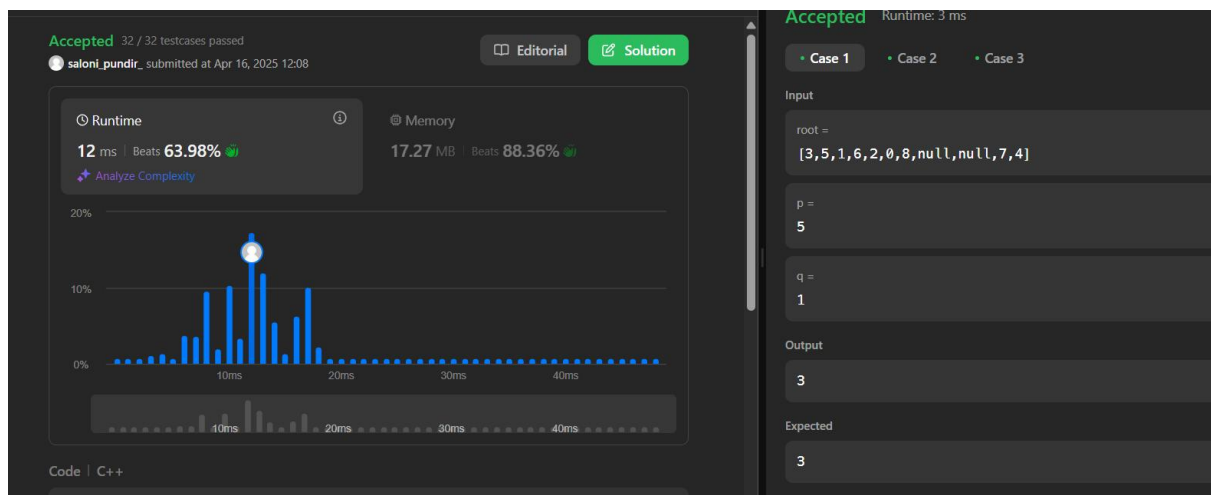
```
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if (root == nullptr || root == p || root == q) {
            return root;
        }

        TreeNode* left = lowestCommonAncestor(root->left, p, q);
        TreeNode* right = lowestCommonAncestor(root->right, p, q);

        if (left != nullptr && right != nullptr) {
            return root;
        }

        return left != nullptr ? left : right;
    }
};
```

Output:



Assignment

Q1.Set Matrix Zeroes.

Code:

```
C++ Auto

1 class Solution {
2 public:
3     void setZeroes(vector<vector<int>>& matrix) { int n = matrix.size();
4         int m = matrix[0].size();
5         vector<vector<int>>v(n,vector<int>(m,0));
6         for(int i=0;i<n;i++){
7             for(int j=0;j<m;j++){
8                 if(matrix[i][j]==0){
9                     for(int k=0;k<n;k++){
10                        v[k][j]=-1;
11                    }
12                    for(int k=0;k<m;k++){
13                        v[i][k]=-1;
14                    }
15                }
16            }
17        }
18        for(int i=0;i<n;i++){
19            for(int j=0;j<m;j++){
20                if(v[i][j]==-1)matrix[i][j]=0;
21            }
22        }
23    }
24 };
```

Output:

Accepted 202 / 202 testcases passed

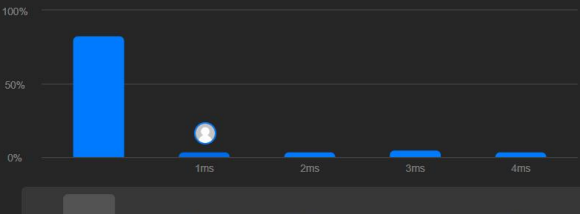
saloni_pundir, submitted at Apr 16, 2025 12:54

Editorial Solution

Runtime 1 ms | Beats 16.80%

Analyze Complexity

Memory 18.98 MB | Beats 6.86%



1ms 2ms 3ms 4ms

Accepted Runtime: 0 ms

Case 1 Case 2

Input

matrix =

```
[[1,1,1],[1,0,1],[1,1,1]]
```

Output

```
[[1,0,1],[0,0,0],[1,0,1]]
```

Expected

```
[[1,0,1],[0,0,0],[1,0,1]]
```

Contribute a testcase

Q2. Longest Substring Without Repeating

Characters.

Code:

```
1  class Solution {
2  public:
3      int lengthOfLongestSubstring(string s) {
4          int n = s.length();
5          int maxLength = 0;
6          unordered_set<char> charSet;
7          int left = 0;
8
9          for (int right = 0; right < n; right++) {
10             if (charSet.count(s[right]) == 0) {
11                 charSet.insert(s[right]);
12                 maxLength = max(maxLength, right - left + 1);
13             } else {
14                 while (charSet.count(s[right])) {
15                     charSet.erase(s[left]);
16                     left++;
17                 }
18                 charSet.insert(s[right]);
19             }
20         }
21
22         return maxLength;
23     }
24 }
```

Output:

← All Submissions

Accepted 987 / 987 testcases passed

saloni_pundir_ submitted at Apr 16, 2025 12:56

Editorial Solution

Runtime 18 ms | Beats 33.69%
Analyze Complexity

Memory 14.39 MB | Beats 23.30%

0.05% of solutions used 305 ms of runtime

Code | C++

```
class Solution {
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s = "abcabcbb"

Output

3

Expected

3

Contribute a testcase

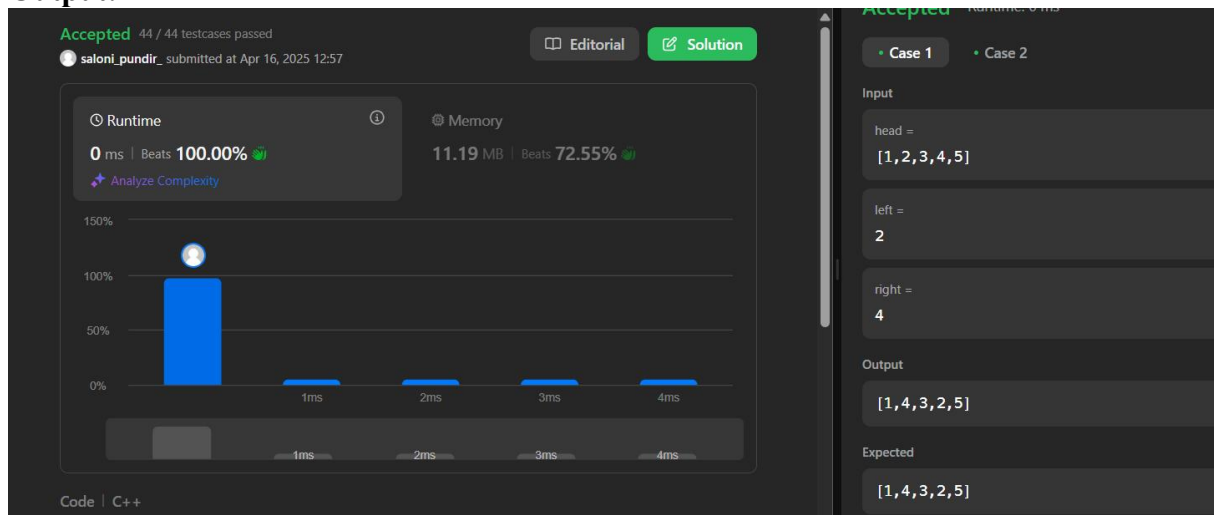
Q3. Reverse Linked List II.

Code:

```
C++ v Auto

1  class Solution {
2  public:
3      ListNode* reverseBetween(ListNode* head, int left, int right) {
4          if (!head || left == right) {
5              return head;
6          }
7
8          ListNode* dummy = new ListNode(0);
9          dummy->next = head;
10         ListNode* prev = dummy;
11
12         for (int i = 0; i < left - 1; i++) {
13             prev = prev->next;
14         }
15
16         ListNode* cur = prev->next;
17
18         for (int i = 0; i < right - left; i++) {
19             ListNode* temp = cur->next;
20             cur->next = temp->next;
21             temp->next = prev->next;
22             prev->next = temp;
23         }
24
25         return dummy->next;
26     }
27 };
```

Output:



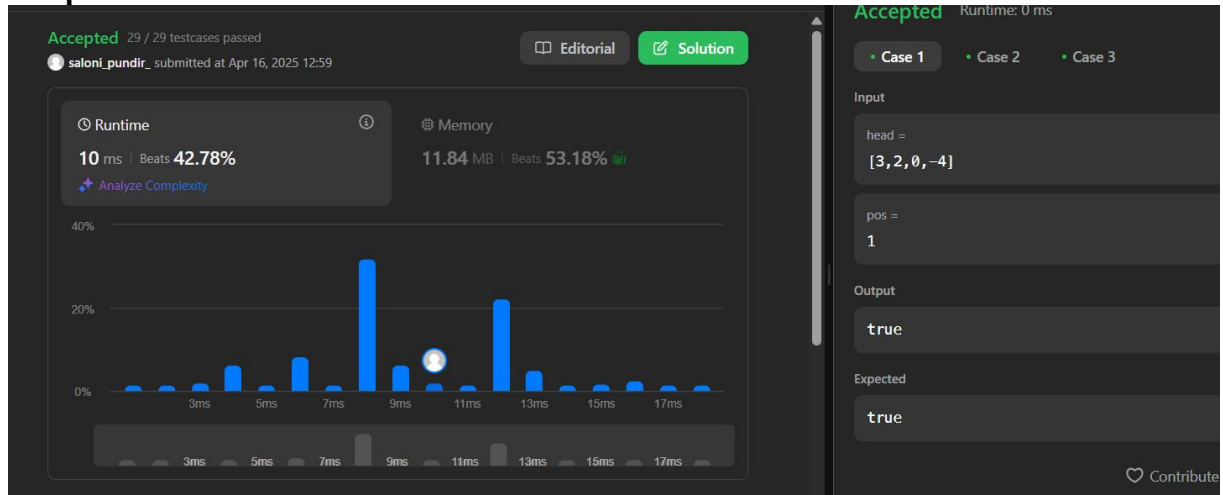
Q4. Detect a Cycle in a Linked

List.

Code:

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode* slow = head;
        ListNode* fast = head;
        while (fast != nullptr && fast->next != nullptr) {
            slow = slow->next;
            fast = fast->next->next;
            if (slow == fast)
                return true;
        }
        return false;
    }
};
```


Output:



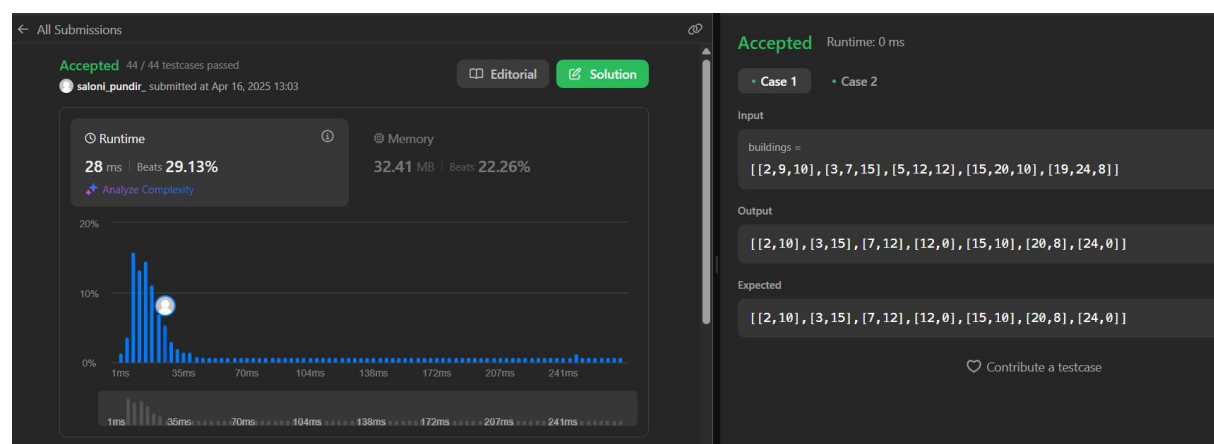
Q5. The Skyline

Problem.

Code:

```
C++ v Auto
1 class Solution {
2 public:
3     vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
4         vector<vector<int>> skyline;
5         map<int, vector<pair<int, int>>> map;
6         for (auto& building : buildings) {
7             map[building[0]].push_back({building[2], 0});
8             map[building[1]].push_back({building[2], 1});
9         }
10        multiset<int> q;
11        for (auto& [pos, heights] : map) {
12            for (auto& [height, type] : heights) {
13                if (type == 0) q.insert(height);
14                else q.erase(q.find(height));
15            }
16            int newHeight = q.empty() ? 0 : *q.rbegin();
17            if (!skyline.empty() && skyline.back()[1] == newHeight) continue;
18            else skyline.push_back(vector<int>({pos, newHeight}));
19        }
20        return skyline;
21    }
22 };
```

Output:

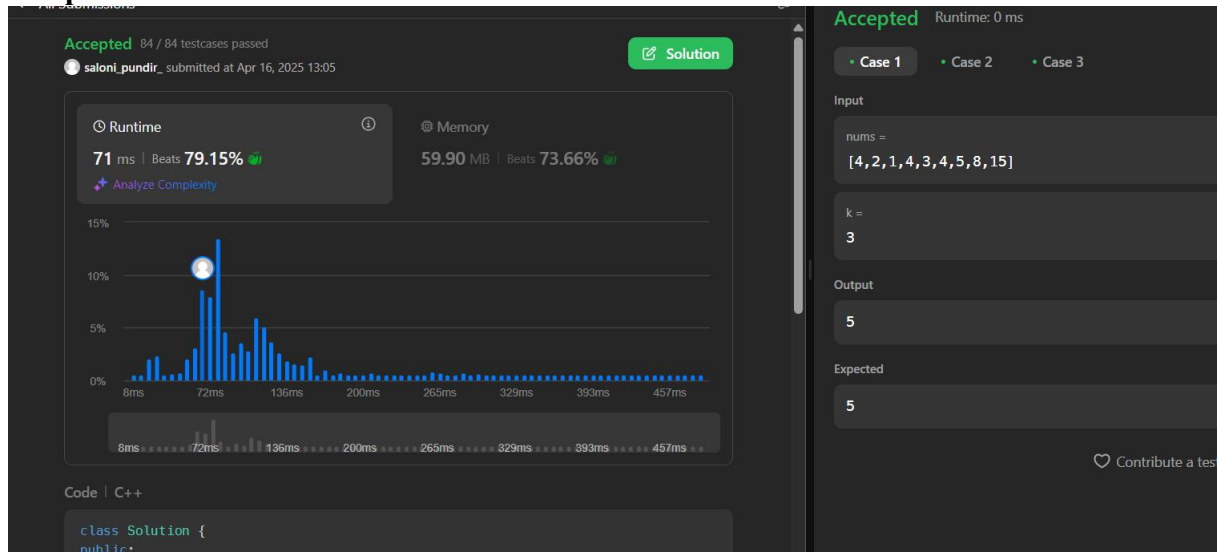


Q6.Longest Increasing Subsequence II

Code:

```
C++ v Auto
1 class Solution {
2 public:
3     vector<int> tree;
4     void update(int node,int st,int end,int i,int val){
5         if(st==end){
6             tree[node]=max(tree[node],val);
7             return;
8         }
9         int mid=(st+end)/2;
10        if(i<=mid){
11            update(node*2,st,mid,i,val);
12        }else{
13            update(node*2+1,mid+1,end,i,val);
14        }
15        tree[node]=max(tree[node*2],tree[node*2+1]);
16    }
17    int query(int node,int st,int end,int x,int y){
18        if(x>end || y<st) return -1e9;
19        if(st>=x && end<=y){
20            return tree[node];
21        }
22        int mid=(st+end)/2;
23        int left=query(2*node,st,mid,x,y);
24        int right=query(2*node+1,mid+1,end,x,y);
25        return max(left,right);
26    }
27    int lengthOfLIS(vector<int>& nums, int k) {
28        int n=nums.size();
29        if(n==1) return 1;
```

Output:

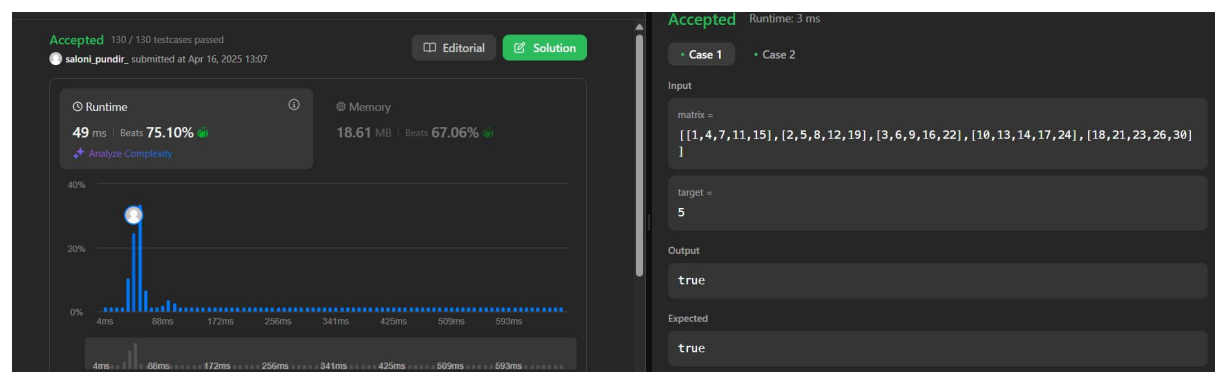


Q7. Search a 2D Matrix II.

Code:

```
C++ Auto  
1 class Solution {  
2 public:  
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {  
4         int m = matrix.size(), n = m ? matrix[0].size() : 0, r = 0, c = n - 1;  
5         while (r < m && c >= 0) {  
6             if (matrix[r][c] == target) {  
7                 return true;  
8             }  
9             matrix[r][c] > target ? c-- : r++;  
10        }  
11        return false;  
12    }  
13 };
```

Output:

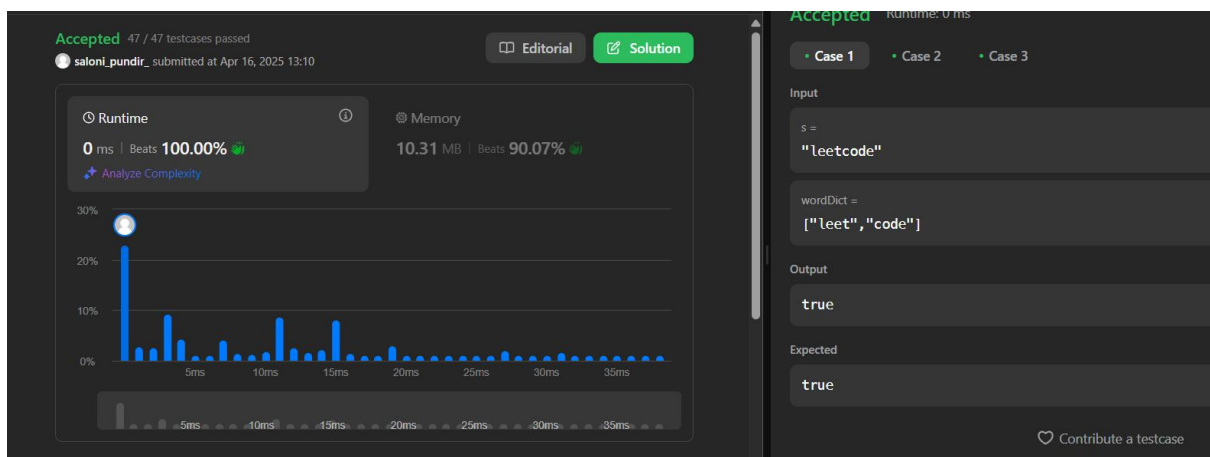


Q8. Word Break.

Code:

```
C++ v Auto
1 class Solution {
2 public:
3     bool wordBreak(string s, vector<string>& wordDict) {
4         vector<bool> dp(s.size() + 1, false);
5         dp[0] = true;
6
7         for (int i = 1; i <= s.size(); i++) {
8             for (const string& w : wordDict) {
9                 int start = i - w.length();
10                if (start >= 0 && dp[start] && s.substr(start, w.length()) == w) {
11                    dp[i] = true;
12                    break;
13                }
14            }
15        }
16        return dp[s.size()];
17    }
18 };|
```

Output:



Q9.Longest Increasing Path in a Matrix.

Code:

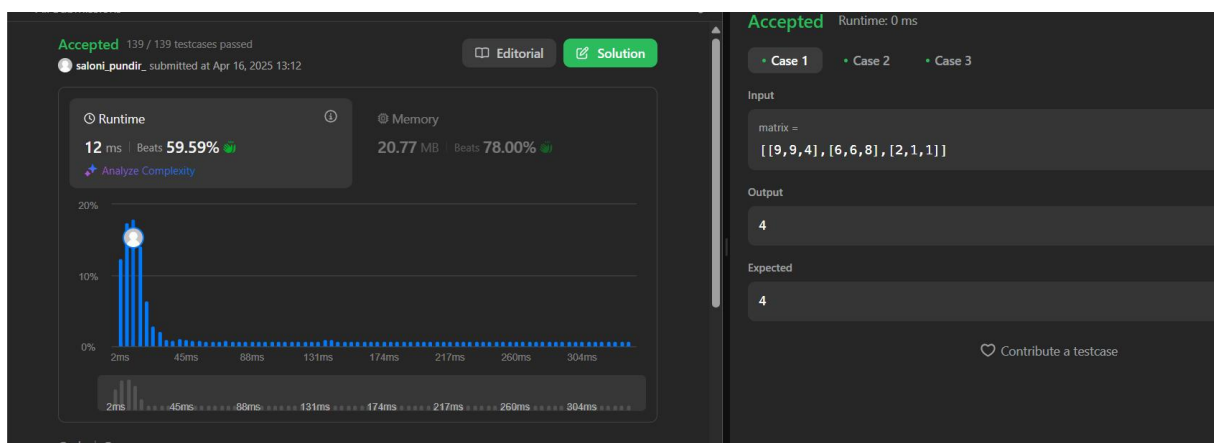
```

C++ v Auto
1 class Solution {
2     vector<vector<int>>> dp;
3     int dr[4] = {-1,0,1,0};
4     int dc[4] = {0,-1,0,1};
5 public:
6     int dfs(int row, int col, vector<vector<int>>>& mat) {
7         int m = mat.size();
8         int n = mat[0].size();
9
10        if (dp[row][col] != 0) {
11            return dp[row][col];
12        }
13
14        int maxLength = 1;
15
16        for (int k=0; k<4; k++) {
17            int nr = row + dr[k];
18            int nc = col + dc[k];
19            if (nr>=0 && nr<m && nc>=0 && nc<n && mat[nr][nc] > mat[row][col]) {
20                maxLength = max(maxLength, 1 + dfs(nr, nc, mat));
21            }
22        }
23
24        return dp[row][col] = maxLength;
25    }
26
27    int longestIncreasingPath(vector<vector<int>>>& matrix) {
28        int m = matrix.size();
29        int n = matrix[0].size();

```

Saved Ln 42,

Output:



Q10.Trapping Rain Water.

Code:

```
C++  Auto

1  class Solution {
2  public:
3      int trap(vector<int>& height) {
4          int left = 0;
5          int right = height.size() - 1;
6          int leftMax = height[left];
7          int rightMax = height[right];
8          int water = 0;
9
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
22         return water;
23     }
24 };
```

Output:

Accepted 324 / 324 testcases passed

saloni_pundir_ submitted at Apr 16, 2025 13:14

Editorial Solution

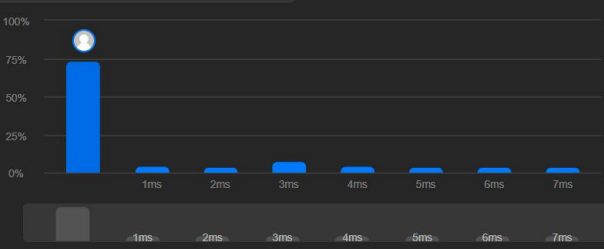
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

26.06 MB | Beats 61.16%



1ms 2ms 3ms 4ms 5ms 6ms 7ms

Accepted Runtime: 0 ms

Case 1 Case 2

Input

height =

[0,1,0,2,1,0,1,3,2,1,2,1]

Output

6

Expected

6

Contribute a testcase