



# DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment-9

**Student Name:** Gaurav Saini

**Branch:** BE-CSE

**Semester:**06

**UID:** 22BCS11085

**Section/Group:**NTPP\_IOT\_603\_B

**Date of Performance:**04-04-2025

**Subject Name:** AP LAB-II

**Subject Code:** 22CSP-351

### 1. Aim:

- a. **Lowest Common Ancestor of a Binary Tree**
- b. **Number of Islands**
- c. **Course Schedule**

### 2. Introduction to Graphs:

- A graph is a data structure that represents relationships between objects using:
- Nodes (Vertices): Represent entities in the graph.
- Edges: Represent connections/relationships between nodes.

#### Types of Nodes

- **Root Node:** The starting point of a graph with no ancestors.
- **Leaf Nodes:** Nodes with no outgoing edges, only incoming connections.



### 3. Implementation/Code:

#### A. Lowest Common Ancestor of a Binary Tree

```
/**  
  
 * Definition for a binary tree node.  
  
 * struct TreeNode {  
  
 *     int val;  
  
 *     TreeNode *left;  
  
 *     TreeNode *right;  
  
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
  
 * };  
  
 */  
  
class Solution {  
  
public:  
  
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p,  
    TreeNode* q) {  
  
        if (root == nullptr || root == p || root == q) {  
  
            return root;  

```



# DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}
```

```
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;
```

```
}
```

```
};
```

## B. Number of Islands

```
class Solution {  
  
public:  
  
    int numIslands(vector<vector<char>>& grid) {  
  
        int m = grid.size();  
  
        int n = grid[0].size();  
  
        int count = 0;  
  
        function<void(int, int)> dfs = [&](int i, int j) {  
  
            if (i < 0 || j < 0 || i >= m || j >= n || grid[i][j] == '0')  
  
                return;  
  
            grid[i][j] = '0';  
  
            dfs(i + 1, j);  
  
            dfs(i - 1, j);  
  
            dfs(i, j + 1);  
  
            dfs(i, j - 1);  
        }  
    }  
};
```



# DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

Discover. Learn. Empower.

```
};
```

```
for (int i = 0; i < m; ++i) {
```

```
    for (int j = 0; j < n; ++j) {
```

```
        if (grid[i][j] == '1') {
```

```
            count++;
```

```
            dfs(i, j);
```

```
        }
```

```
    }
```

```
}
```

```
return count;
```

```
}
```

```
};
```

## C. Course Schedule

```
class Solution {
public:
    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
        vector<vector<int>> adj(numCourses);
        vector<int> indegree(numCourses, 0);

        for (auto& pre : prerequisites) {
            adj[pre[1]].push_back(pre[0]);
            indegree[pre[0]]++;
        }

        queue<int> q;

        for (int i = 0; i < numCourses; i++) {
            if (indegree[i] == 0)
                q.push(i);
        }

        int count = 0;

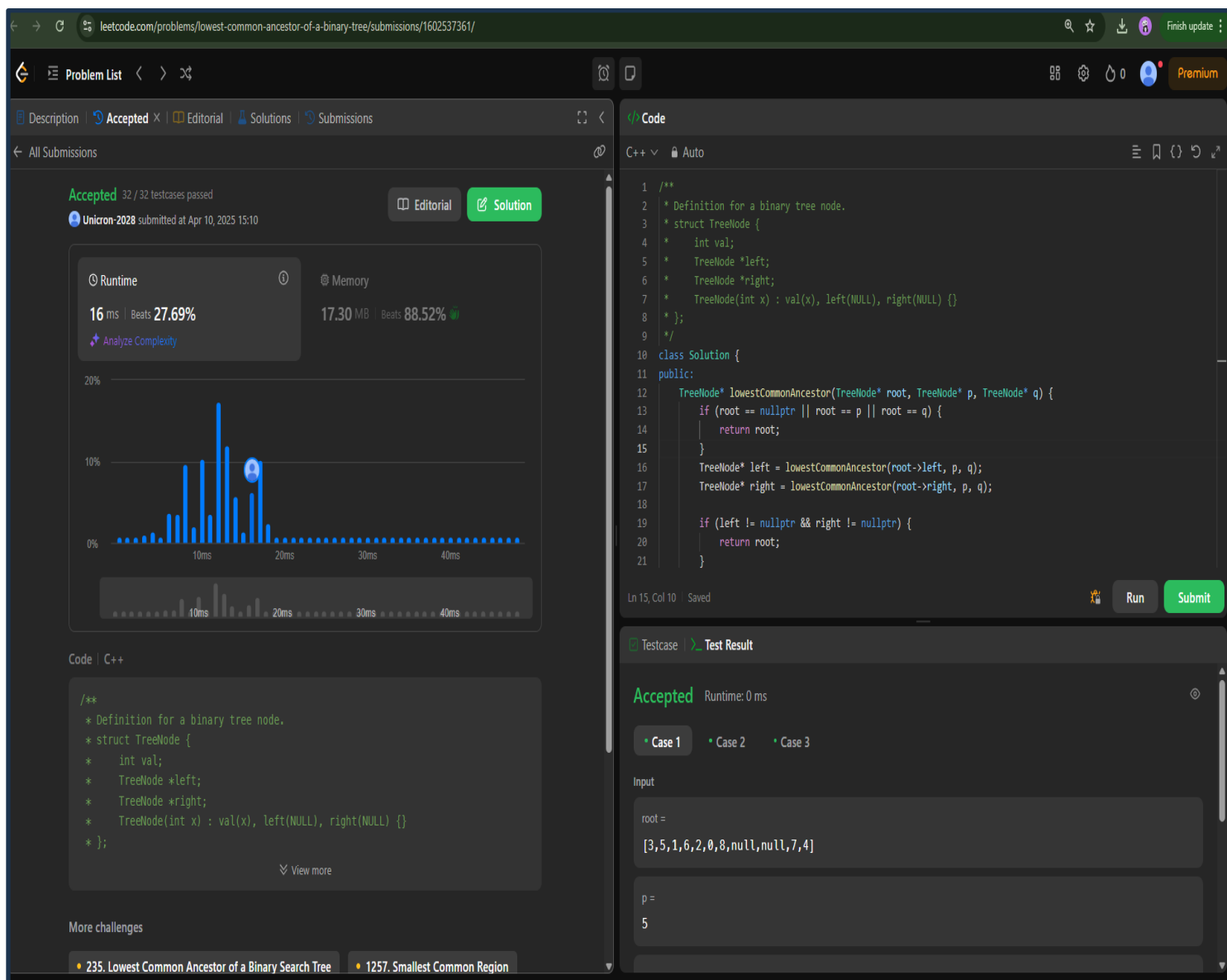
        while (!q.empty()) {
            int curr = q.front();
            q.pop();
            count++;

            for (int next : adj[curr]) {
                indegree[next]--;
                if (indegree[next] == 0)
                    q.push(next);
            }
        }

        return count == numCourses;
    }
};
```

## 4. Output

### A. Lowest Common Ancestor of a Binary Tree



leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/submissions/1602537361/

Problem List < > >>

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 32 / 32 testcases passed

Unicon-2028 submitted at Apr 10, 2025 15:10

Editorial Solution

Runtime 16 ms | Beats 27.69%

Memory 17.30 MB | Beats 88.52%

Analyze Complexity

Code | C++

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
```

View more

More challenges

235. Lowest Common Ancestor of a Binary Search Tree 1257. Smallest Common Region

Code C++ Auto

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8  * };
9  */
10 class Solution {
11 public:
12     TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
13         if (root == nullptr || root == p || root == q) {
14             return root;
15         }
16         TreeNode* left = lowestCommonAncestor(root->left, p, q);
17         TreeNode* right = lowestCommonAncestor(root->right, p, q);
18
19         if (left != nullptr && right != nullptr) {
20             return root;
21         }
22     }
23 }
```

Ln 15, Col 10 Saved Run Submit

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

root =

[3,5,1,6,2,0,8,null,null,7,4]

p =

5



# DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

Discover. Learn. Empower.

## B. Number Of Islands

[Description](#) | [Accepted](#) × | [Editorial](#) | [Solutions](#) | [Submissions](#)

← All Submissions [@](#)

**Accepted** 49 / 49 testcases passed

**Unicron-2028** submitted at Apr 10, 2025 15:27

[Editorial](#) [Solution](#)

**Runtime** ⓘ

26 ms | Beats **64.80%** 🌱

[Analyze Complexity](#)

**Memory**

17.25 MB | Beats **53.64%** 🌱

**Code** | C++

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        int m = grid.size();
        int n = grid[0].size();
        int count = 0;

        function<void(int, int)> dfs = [&](int i, int j) {
```

⌵ View more

More challenges

[130. Surrounded Regions](#) [286. Walls and Gates](#) [305. Number of Islands II](#)





# DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

Discover. Learn. Empower.

## C. Course Schedule

Problem List

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 54 / 54 testcases passed

Unicron-2028 submitted at Apr 10, 2025 15:32

Editorial

Solution

Runtime

3 ms | Beats 81.50%

Analyze Complexity

Memory

19.26 MB | Beats 77.51%

Analyze Complexity

Code | C++

```
class Solution {
public:
    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
        vector<vector<int>> adj(numCourses);
        vector<int> indegree(numCourses, 0);

        for (auto& pre : prerequisites) {
            adj[pre[1]].push_back(pre[0]);
        }
    }
};
```

View more

[More challenges](#)

## 5. Learning Outcomes:

### ➤ Graph Representation:

- Learn how to represent a graph using adjacency list or matrix.
- Understand the direction of edges in directed vs undirected graphs.

### ➤ DFS (Depth-First Search):

- Apply recursive DFS to explore connected components (e.g., islands).
- Use DFS for marking visited nodes and exploring all reachable areas from a given node.

### ➤ BFS (Breadth-First Search):

- Implement BFS for topological sorting (Kahn's Algorithm).
- Learn how to manage in-degree of nodes to determine processing order.

### ➤ Cycle Detection in Graphs:

- Use DFS or BFS (Kahn's) to detect cycles in directed graphs.
- Understand the importance of detecting cycles in scheduling/dependency problems.