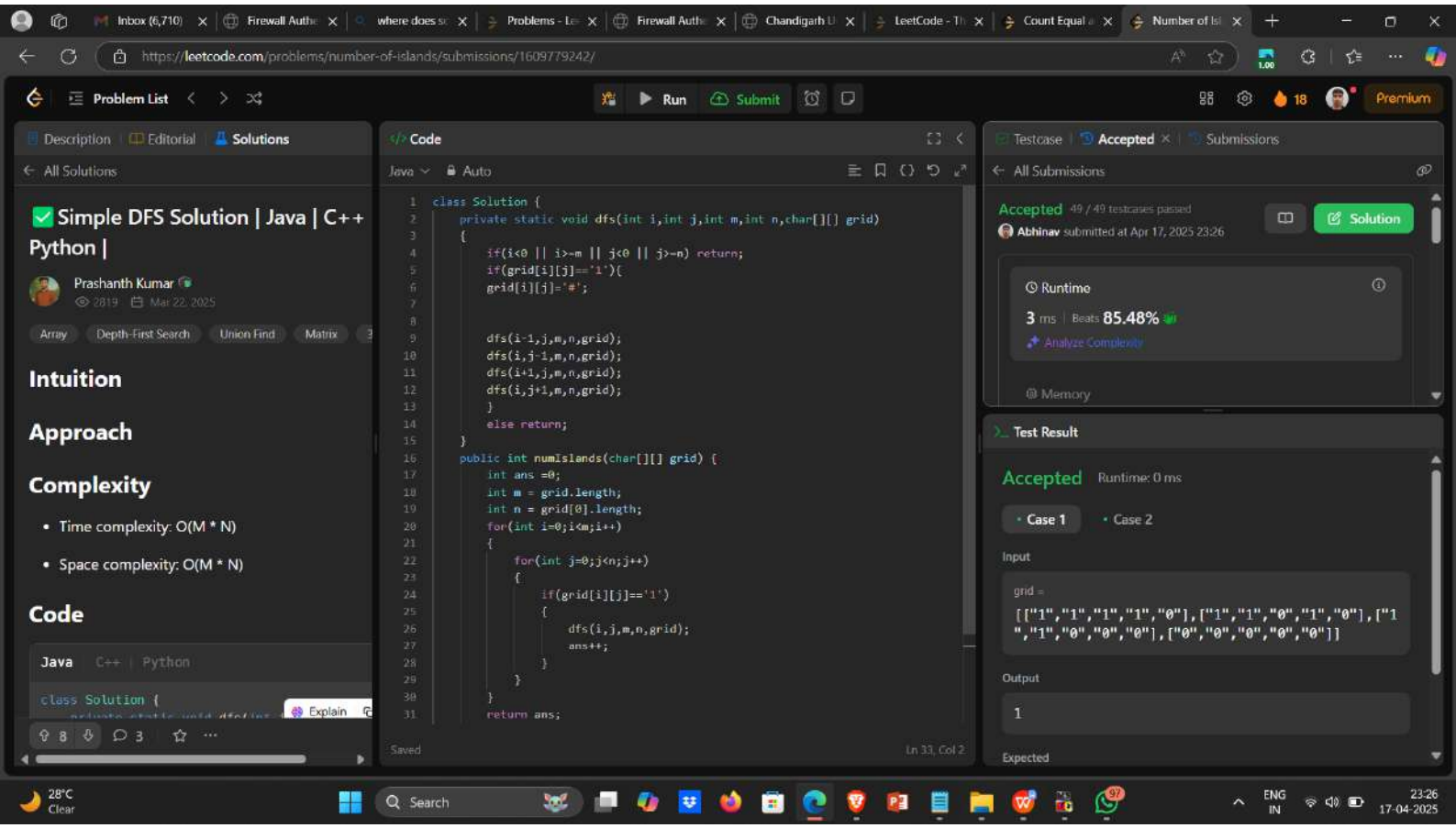
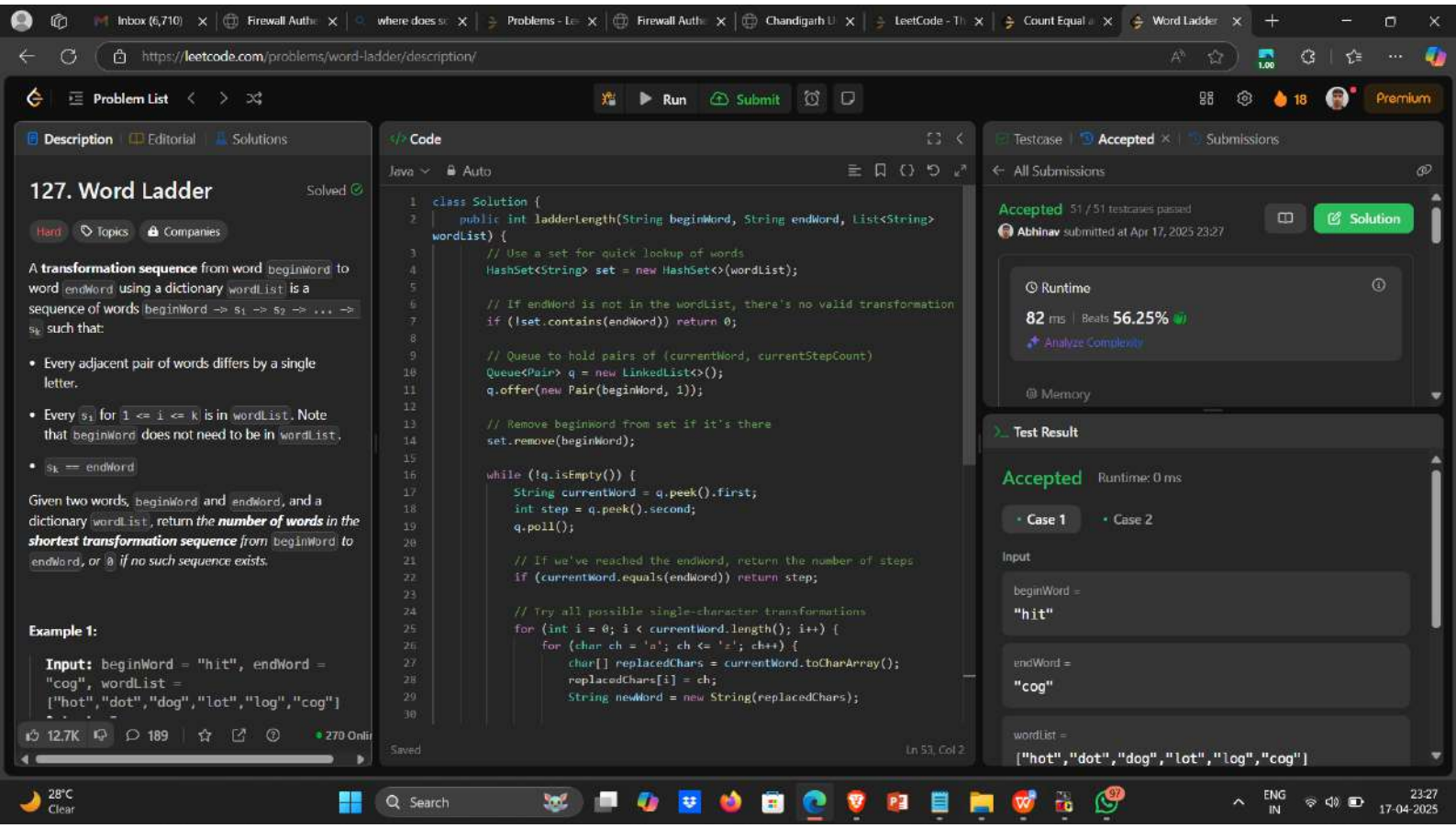
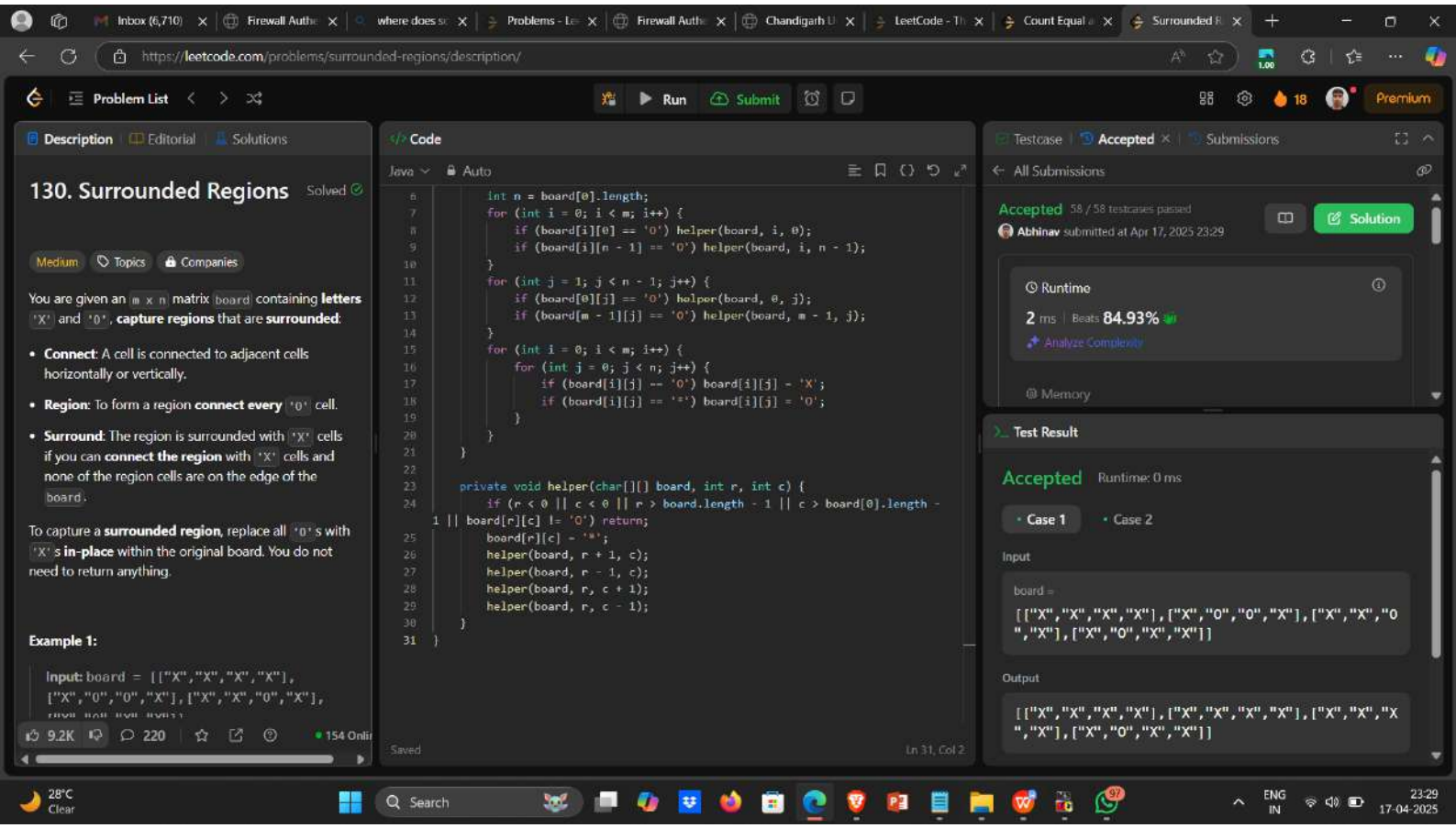

Problem List
Run
Submit
Premium
Description
Editorial
Solutions
All Solutions
Simple DFS Solution | Java | C++ Python |
Prashanth Kumar
2819
Mar 22, 2025
Array
Depth-First Search
Union Find
Matrix
Intuition
Approach
Complexity
Time complexity: O(M * N)
Space complexity: O(M * N)
Code
Java
C++
Python
Explain
Code
Java
Auto
class Solution {
private static void dfs(int i,int j,int m,int n,char[][] grid)
{
if(i<0 || i>=m || j<0 || j>=n) return;
if(grid[i][j]=='1'){
grid[i][j]='#';
dfs(i-1,j,m,n,grid);
dfs(i,j-1,m,n,grid);
dfs(i+1,j,m,n,grid);
dfs(i,j+1,m,n,grid);
}
else return;
}
public int numIslands(char[][] grid) {
int ans =0;
int m = grid.length;
int n = grid[0].length;
for(int i=0;i<m;i++)
{
for(int j=0;j<n;j++)
{
if(grid[i][j]=='1')
{
dfs(i,j,m,n,grid);
ans++;
}
}
}
return ans;
Saved
Ln 33, Col 2
Testcase
Accepted
Submissions
All Submissions
Accepted
49 / 49 testcases passed
Abhinav submitted at Apr 17, 2025 23:26
Solution
Runtime
3 ms
Beats 85.48%
Analyze Complexity
Memory
Test Result
Accepted
Runtime: 0 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


Inbox (6,710)
Firewall Authe
where does s
Problems - Le
Firewall Authe
Chandigarh U
LeetCode - Th
Count Equal
Word Ladder
https://leetcode.com/problems/word-ladder/description/
Problem List
Run
Submit
18
Premium
Description
Editorial
Solutions
127. Word Ladder
Solved
Hard
Topics
Companies
A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:
Every adjacent pair of words differs by a single letter.
Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
sk == endWord
Given two words, beginWord and endWord, and a dictionary wordList, return the number of words in the shortest transformation sequence from beginWord to endWord, or 0 if no such sequence exists.
Example 1:
Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
12.7K
189
270 Onli
Code
Java
Auto
class Solution {
public int ladderLength(String beginWord, String endWord, List<String> wordList) {
// Use a set for quick lookup of words
HashSet<String> set = new HashSet<>(wordList);
// If endWord is not in the wordList, there's no valid transformation
if (!set.contains(endWord)) return 0;
// Queue to hold pairs of (currentWord, currentStepCount)
Queue<Pair> q = new LinkedList<>();
q.offer(new Pair(beginWord, 1));
// Remove beginWord from set if it's there
set.remove(beginWord);
while (!q.isEmpty()) {
String currentWord = q.peek().first;
int step = q.peek().second;
q.poll();
// If we've reached the endWord, return the number of steps
if (currentWord.equals(endWord)) return step;
// Try all possible single-character transformations
for (int i = 0; i < currentWord.length(); i++) {
for (char ch = 'a'; ch <= 'z'; ch++) {
char[] replacedChars = currentWord.toCharArray();
replacedChars[i] = ch;
String newWord = new String(replacedChars);
Saved
Ln 53, Col 2
Testcase
Accepted
Submissions
All Submissions
Accepted 51 / 51 testcases passed
Abhinav submitted at Apr 17, 2025 23:27
Solution
Runtime
82 ms | Beats 56.25%
Analyze Complexity
Memory
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Input
beginWord =
"hit"
endWord =
"cog"
wordList =
["hot","dot","dog","lot","log","cog"]
28°C
Clear
Search
ENG
IN
23:27
17-04-2025


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
130. Surrounded Regions
Solved
Medium
Topics
Companies
You are given an m x n matrix board containing letters 'X' and 'O', capture regions that are surrounded:
Connect: A cell is connected to adjacent cells horizontally or vertically.
Region: To form a region connect every 'O' cell.
Surround: The region is surrounded with 'X' cells if you can connect the region with 'X' cells and none of the region cells are on the edge of the board.
To capture a surrounded region, replace all 'O's with 'X's in-place within the original board. You do not need to return anything.
Example 1:
Input: board = [["X","X","X","X"],
["X","O","O","X"],["X","X","O","X"],
9.2K
220
154 Onli
Code
Java
Auto
int n = board[0].length;
for (int i = 0; i < m; i++) {
if (board[i][0] == 'O') helper(board, i, 0);
if (board[i][n - 1] == 'O') helper(board, i, n - 1);
}
for (int j = 1; j < n - 1; j++) {
if (board[0][j] == 'O') helper(board, 0, j);
if (board[m - 1][j] == 'O') helper(board, m - 1, j);
}
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
if (board[i][j] == 'O') board[i][j] = 'X';
if (board[i][j] == '*') board[i][j] = 'O';
}
}
}
private void helper(char[][] board, int r, int c) {
if (r < 0 || c < 0 || r > board.length - 1 || c > board[0].length - 1 || board[r][c] != 'O') return;
board[r][c] = '*';
helper(board, r + 1, c);
helper(board, r - 1, c);
helper(board, r, c + 1);
helper(board, r, c - 1);
}
}
Saved
Ln 31, Col 2
Testcase
Accepted
Submissions
All Submissions
Accepted
58 / 58 testcases passed
Abhinav submitted at Apr 17, 2025 23:29
Solution
Runtime
2 ms | Beats 84.93%
Analyze Complexity
Memory
Test Result
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"]]
28°C
Clear
Search
ENG
IN
23:29
17-04-2025

https://leetcode.com/problems/binary-tree-maximum-path-sum/submissions/1609783260/

Problem List | Run | Submit | 18 | Premium

Description | Editorial | Solutions

124. Binary Tree Maximum Path Sum

Solved

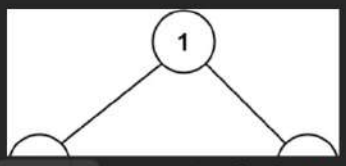
Hard | Topics | Companies

A **path** in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence **at most once**. Note that the path does not need to pass through the root.

The **path sum** of a path is the sum of the node's values in the path.

Given the `root` of a binary tree, return *the maximum **path sum** of any **non-empty** path.*

Example 1:



17.4K | 241 | 255 Onlir

Code

Java | Auto

```java
class Solution {
    private int maxSum = Integer.MIN_VALUE;

    public int maxPathSum(TreeNode root) {
        helper(root);
        return maxSum;
    }

    private int helper(TreeNode node) {
        if (node == null) {
            return 0;
        }

        int leftMaxPath = Math.max(helper(node.left), 0);
        int rightMaxPath = Math.max(helper(node.right), 0);

        int maxIfNodeIsRoot = node.val + leftMaxPath + rightMaxPath;
        maxSum = Math.max(maxSum, maxIfNodeIsRoot);

        return node.val + Math.max(leftMaxPath, rightMaxPath);
    }
}
```

Saved | Ln 22, Col 2

Testcase | Accepted | Submissions

All Submissions

Accepted 96 / 96 testcases passed

Abhinav submitted at Apr 17, 2025 23:30

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[1,2,3]

Output

6

Expected

28°C Clear | Search | ENG IN | 23:30 17-04-2025

Inbox (6,710) | Firewall Authe | where does s | Problems - Le | Firewall Authe | Chandigarh U | LeetCode - Th | Count Equal e | Number of Pr

https://leetcode.com/problems/number-of-provinces/submissions/1609784128/

Problem List | Run | Submit | 18 | Premium

Description | Editorial | Solutions

547. Number of Provinces

Solved

Medium | Topics | Companies

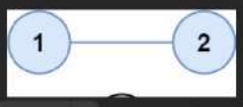
There are `n` cities. Some of them are connected, while some are not. If city `a` is connected directly with city `b`, and city `b` is connected directly with city `c`, then city `a` is connected indirectly with city `c`.

A **province** is a group of directly or indirectly connected cities and no other cities outside of the group.

You are given an `n x n` matrix `isConnected` where `isConnected[i][j] = 1` if the i^{th} city and the j^{th} city are directly connected, and `isConnected[i][j] = 0` otherwise.

Return *the total number of* ***provinces***.

Example 1:



10.3K | 135 | 145 Onlir

Code

Java | Auto

```java
class Solution {
    boolean visited[];
    public int findCircleNum(int[][] isConnected) {
        visited=new boolean[isConnected.length];
        int cnt=0;
        for(int i=0;i<visited.length;i++)
        {
            if(!visited[i])
            {
                dfs(isConnected, i);
                cnt++;
            }
        }
        return cnt;
    }

    private void dfs(int[][] isConnected, int curr)
    {
        visited[curr]=true;
        for(int i=0;i<isConnected[curr].length;i++)
        {
            if(isConnected[curr][i]==1 && !visited[i]) dfs(isConnected, i);
        }
    }
}
```

Saved | Ln 25, Col 2

Testcase | Accepted | Submissions

All Submissions

Accepted 114 / 114 testcases passed

Abhinav submitted at Apr 17, 2025 23:31

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

isConnected =

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

Output

2

Expected

28°C Clear | Search | ENG IN | 23:31 17-04-2025

https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/submissions/1609785611/

Problem List | Run | Submit | 18 | Premium

Description | Editorial | Solutions

236. Lowest Common Ancestor of a Binary Tree

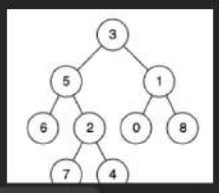
Solved

Medium | Topics | Companies

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**)."

Example 1:



17.5K | 124 | 285 Onlir

Code

Java | Auto

```java
public class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if(root == null || root == p || root == q)  return root;
        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p, q);
        if(left != null && right != null)   return root;
        return left != null ? left : right;
    }
}
```

Saved | Ln 9, Col 2

Testcase | Accepted | Submissions

All Submissions

Accepted 32 / 32 testcases passed

Abhinav submitted at Apr 17, 2025 23:33

Solution

Runtime

6 ms | Beats 100.00%

Analyze Complexity

Memory

Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

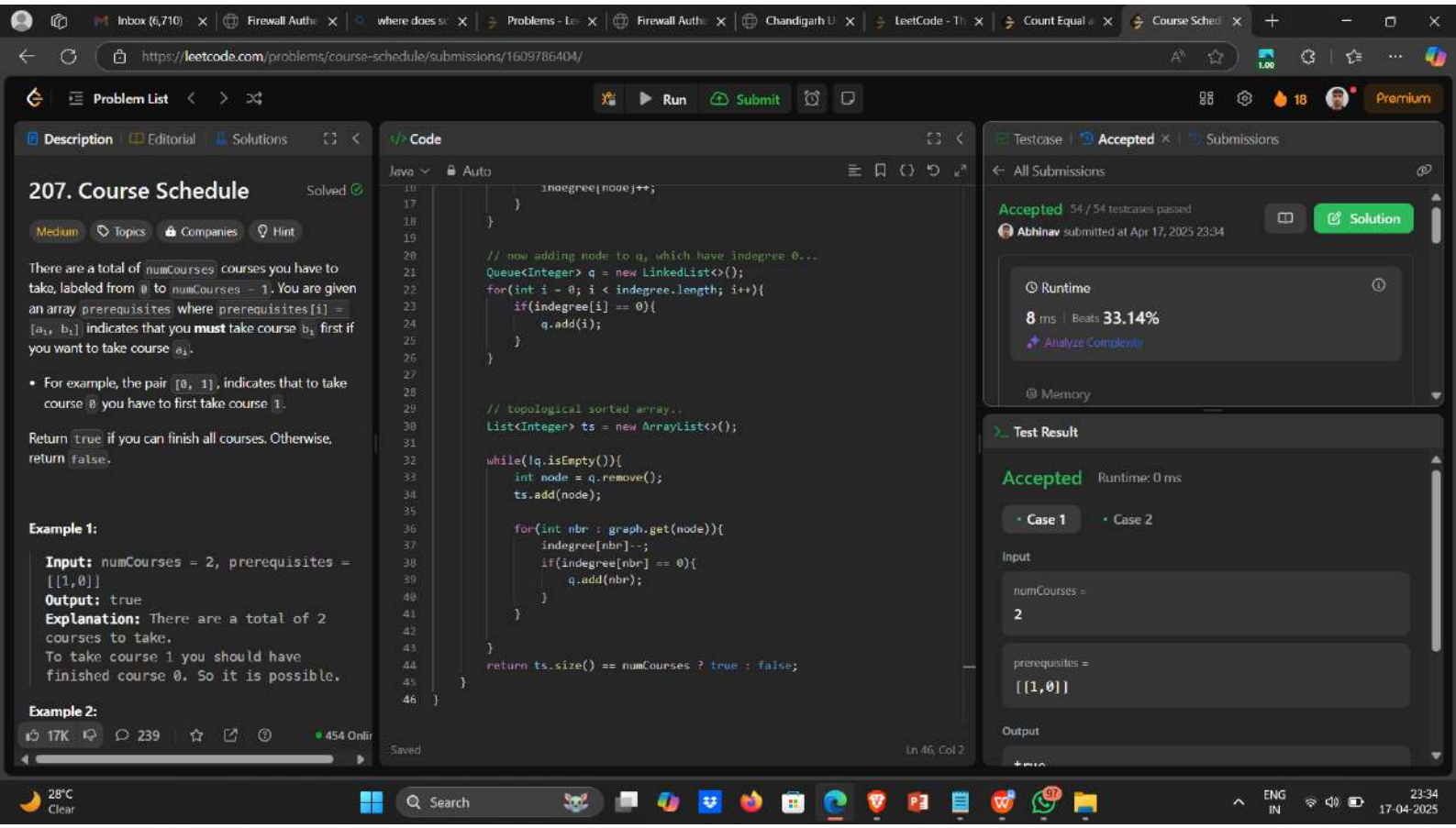
Input

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

p =
5

q =
1

28°C Clear | Search | ENG IN | 23:33 17-04-2025


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
207. Course Schedule
Solved
Medium
Topics
Companies
Hint
There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [a_i, b_i] indicates that you must take course b_i first if you want to take course a_i.
For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.
Return true if you can finish all courses. Otherwise, return false.
Example 1:
Input: numCourses = 2, prerequisites = [[1,0]]
Output: true
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0. So it is possible.
Example 2:
17K
239
454 Onlir
Code
Java
Auto
indegree[node]++;
}
}
// now adding node to q, which have indegree 0...
Queue<Integer> q = new LinkedList<>();
for(int i = 0; i < indegree.length; i++){
if(indegree[i] == 0){
q.add(i);
}
}
// topological sorted array..
List<Integer> ts = new ArrayList<>();
while(!q.isEmpty()){
int node = q.remove();
ts.add(node);
for(int nbr : graph.get(node)){
indegree[nbr]--;
if(indegree[nbr] == 0){
q.add(nbr);
}
}
}
return ts.size() == numCourses ? true : false;
}
}
Saved
Ln 46, Col 2
Testcase
Accepted
Submissions
All Submissions
Accepted
54 / 54 testcases passed
Abhinav submitted at Apr 17, 2025 23:34
Solution
Runtime
8 ms | Beats 33.14%
Analyze Complexity
Memory
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
numCourses =
2
prerequisites =
[[1,0]]
Output

https://leetcode.com/problems/longest-increasing-path-in-a-matrix/submissions/1609787184/

Problem List | Run | Submit | Premium

Description | Editorial | Solutions

329. Longest Increasing Path in a Matrix

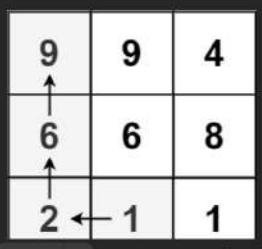
Solved

Hard | Topics | Companies

Given an `m x n` integers `matrix`, return *the length of the longest increasing path in* `matrix`.

From each cell, you can either move in four directions: left, right, up, or down. You **may not** move **diagonally** or move **outside the boundary** (i.e., wrap-around is not allowed).

Example 1:



9.2K | 68 | 67 Onlir

Code

Java | Auto

```java
                }
            }
        }

        int longestPath = 0;

        // Perform BFS with topological sorting
        while (!queue.isEmpty()) {
            int size = queue.size();
            longestPath++; // Increment path length for each level processed
            for (int i = 0; i < size; i++) {
                int[] cell = queue.poll();
                int row = cell[0], col = cell[1];

                for (int[] direction : directions) {
                    int x = row + direction[0];
                    int y = col + direction[1];
                    if (x >= 0 && x < m && y >= 0 && y < n && matrix[x][y] > matrix[row][col]) {
                        indegree[x][y]--;
                        if (indegree[x][y] == 0) {
                            queue.offer(new int[]{x, y});
                        }
                    }
                }
            }
        }

        return longestPath;
    }
}
```

Saved | Ln 59, Col 2

Testcase | Accepted | Submissions

All Submissions

Accepted 139 / 139 testcases passed

Abhinav submitted at Apr 17, 2025 23:34

Solution

Runtime

16 ms | Beats 17.19%

Analyze Complexity

Memory

Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

matrix =

[[9,9,4],[6,6,8],[2,1,1]]

Output

4

Expected

28°C Clear | Search | ENG IN | 23:35 17-04-2025