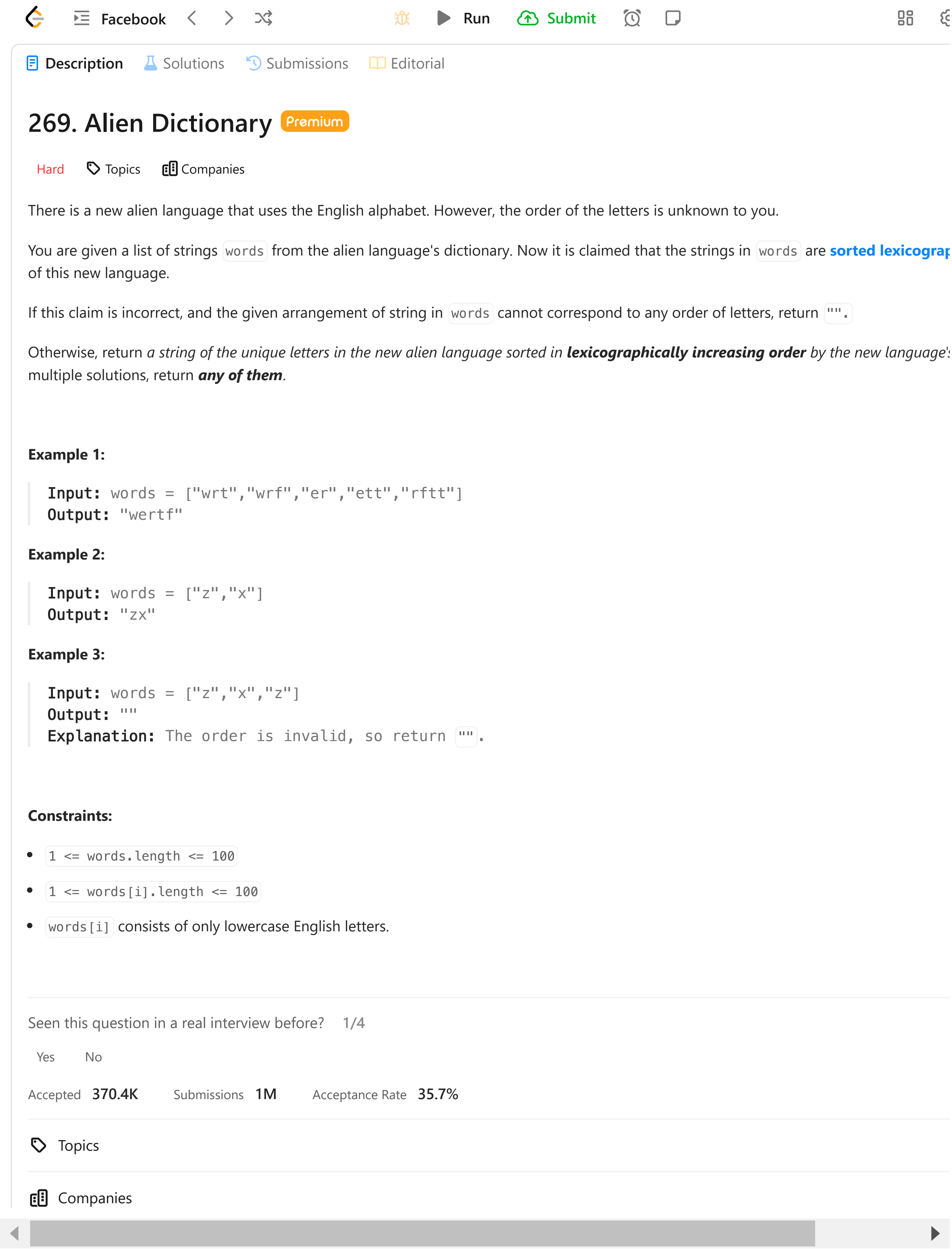


269. Alien Dictionary

There is a new alien language that uses the English alphabet. However, the order of the letters is unknown to you.

You are given a list of strings `words` from the alien language's dictionary. Now it is claimed that the strings in `words` are **sorted lexicograp**
of this new language.

If this claim is incorrect, and the given arrangement of string in `words` cannot correspond to any order of letters, return `""`.

Otherwise, return *a string of the unique letters in the new alien language sorted in* ***lexicographically increasing order*** *by the new language'*
multiple solutions, return ***any of them***.

Example 1:

```
Input: words = ["wrt","wrf","er","ett","rftt"]
Output: "wertf"
```

Example 2:

```
Input: words = ["z","x"]
Output: "zx"
```

Example 3:

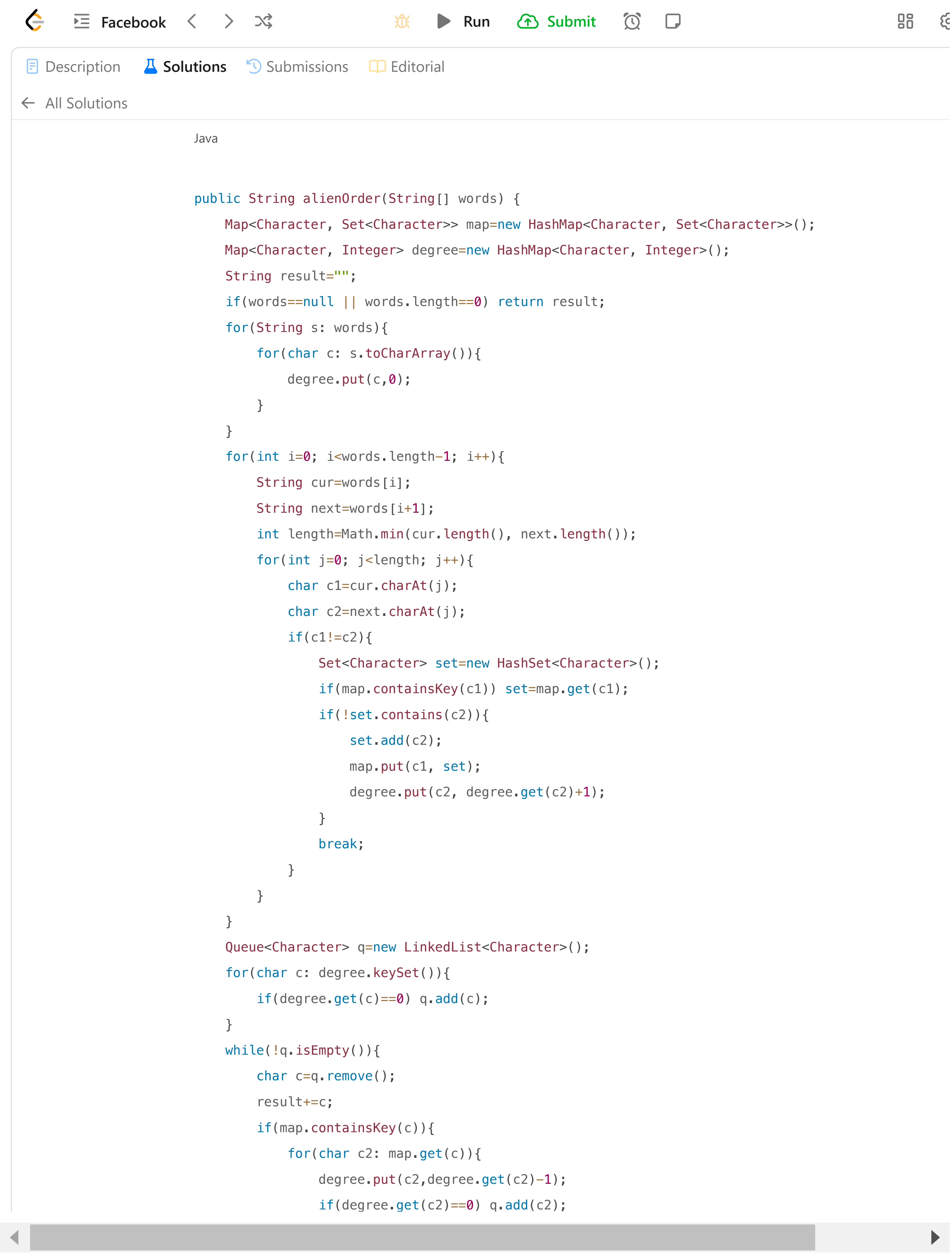
```
Input: words = ["z","x","z"]
Output: ""
Explanation: The order is invalid, so return "".
```

Constraints:

- `1 <= words.length <= 100`
- `1 <= words[i].length <= 100`
- `words[i]` consists of only lowercase English letters.

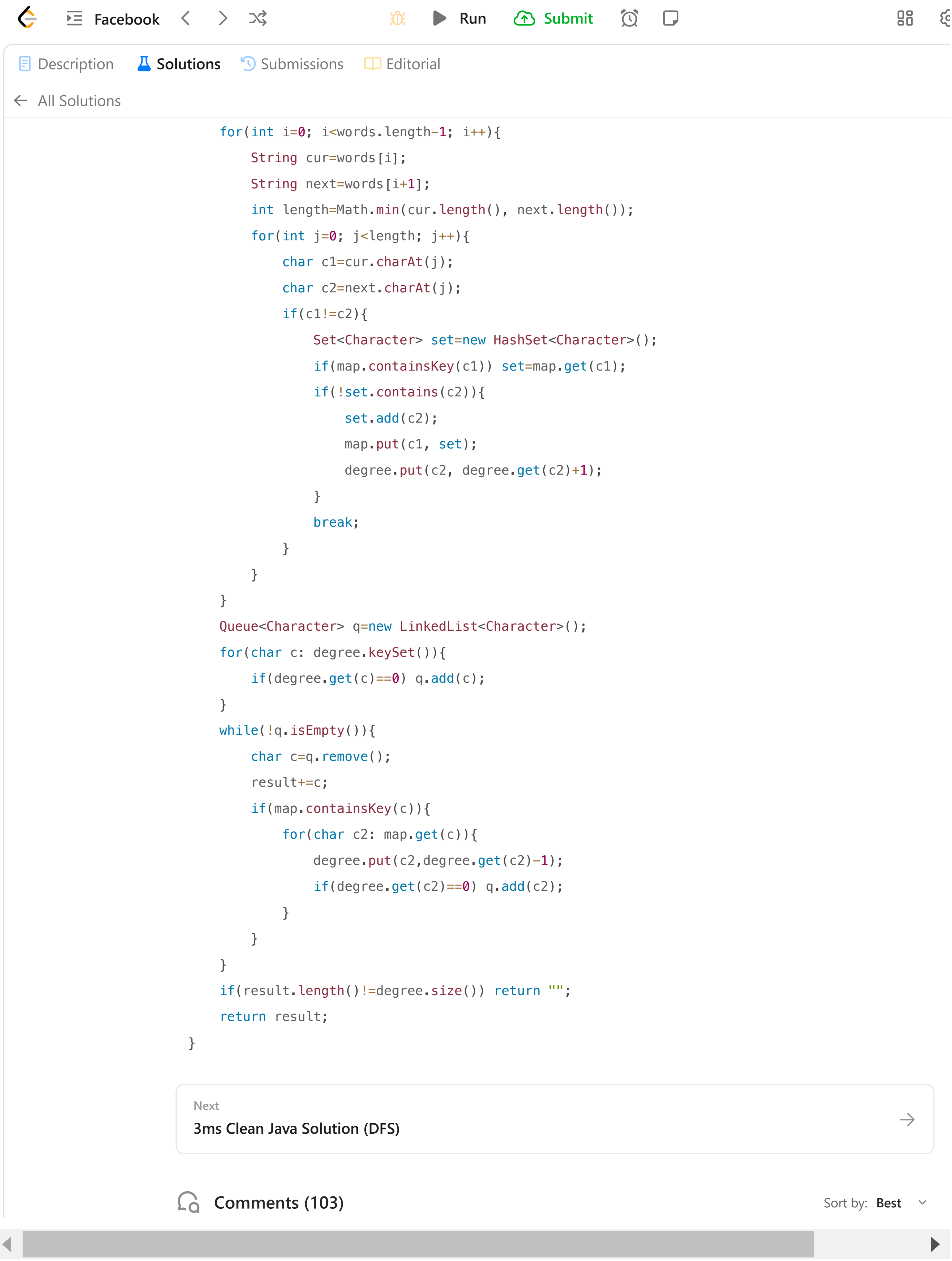
Seen this question in a real interview before? 1/4

Accepted 370.4K Submissions 1M Acceptance Rate 35.7%

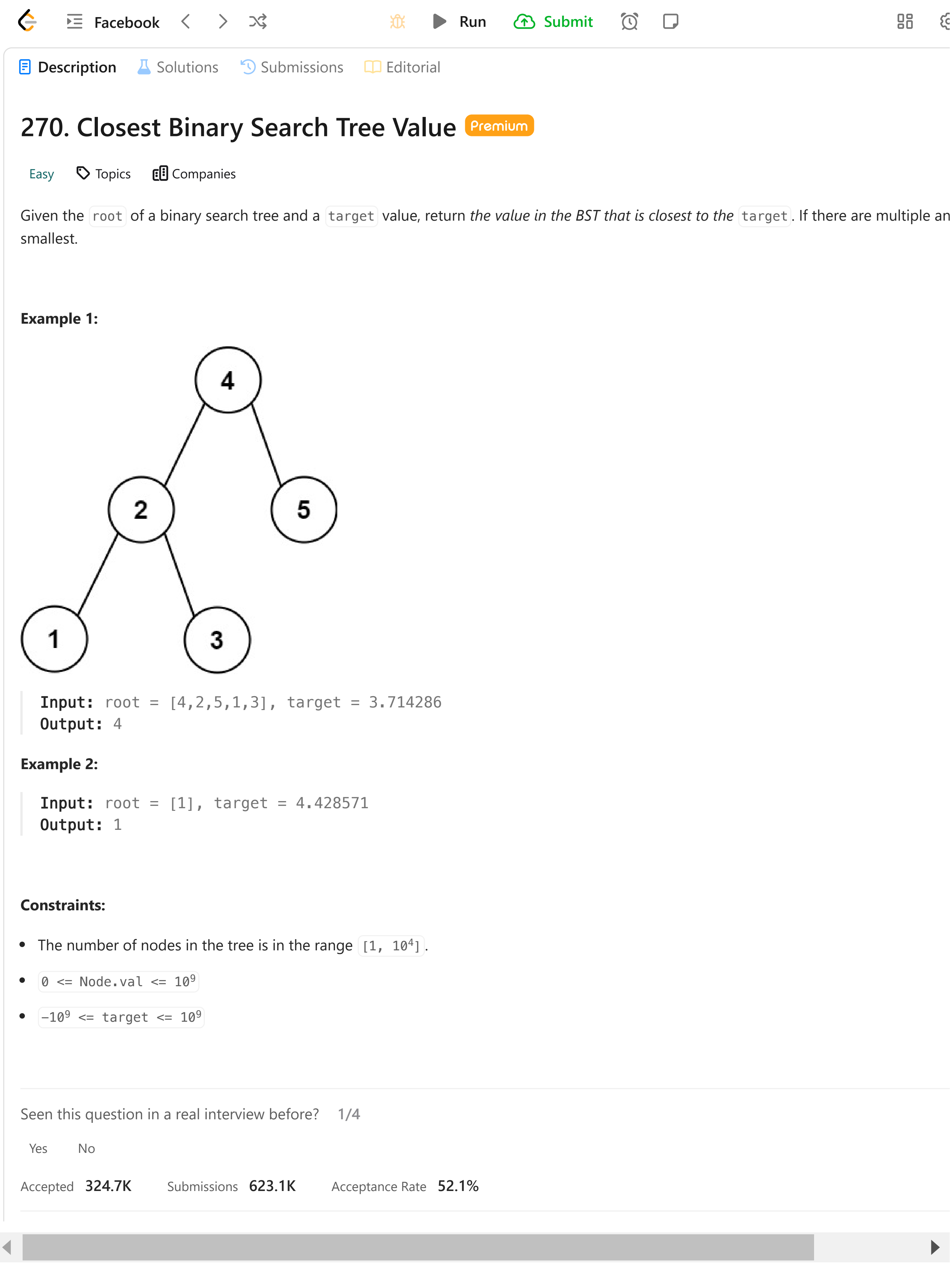


Java

```
public String alienOrder(String[] words) {
    Map<Character, Set<Character>> map=new HashMap<Character, Set<Character>>();
    Map<Character, Integer> degree=new HashMap<Character, Integer>();
    String result="";
    if(words==null || words.length==0) return result;
    for(String s: words){
        for(char c: s.toCharArray()){
            degree.put(c,0);
        }
    }
    for(int i=0; i<words.length-1; i++){
        String cur=words[i];
        String next=words[i+1];
        int length=Math.min(cur.length(), next.length());
        for(int j=0; j<length; j++){
            char c1=cur.charAt(j);
            char c2=next.charAt(j);
            if(c1!=c2){
                Set<Character> set=new HashSet<Character>();
                if(map.containsKey(c1)) set=map.get(c1);
                if(!set.contains(c2)){
                    set.add(c2);
                    map.put(c1, set);
                    degree.put(c2, degree.get(c2)+1);
                }
                break;
            }
        }
    }
    Queue<Character> q=new LinkedList<Character>();
    for(char c: degree.keySet()){
        if(degree.get(c)==0) q.add(c);
    }
    while(!q.isEmpty()){
        char c=q.remove();
        result+=c;
        if(map.containsKey(c)){
            for(char c2: map.get(c)){
                degree.put(c2,degree.get(c2)-1);
                if(degree.get(c2)==0) q.add(c2);
```



```
        for(int i=0; i<words.length-1; i++){
            String cur=words[i];
            String next=words[i+1];
            int length=Math.min(cur.length(), next.length());
            for(int j=0; j<length; j++){
                char c1=cur.charAt(j);
                char c2=next.charAt(j);
                if(c1!=c2){
                    Set<Character> set=new HashSet<Character>();
                    if(map.containsKey(c1)) set=map.get(c1);
                    if(!set.contains(c2)){
                        set.add(c2);
                        map.put(c1, set);
                        degree.put(c2, degree.get(c2)+1);
                    }
                    break;
                }
            }
        }
        Queue<Character> q=new LinkedList<Character>();
        for(char c: degree.keySet()){
            if(degree.get(c)==0) q.add(c);
        }
        while(!q.isEmpty()){
            char c=q.remove();
            result+=c;
            if(map.containsKey(c)){
                for(char c2: map.get(c)){
                    degree.put(c2,degree.get(c2)-1);
                    if(degree.get(c2)==0) q.add(c2);
                }
            }
        }
        if(result.length()!=degree.size()) return "";
        return result;
    }
```



270. Closest Binary Search Tree Value

Given the `root` of a binary search tree and a `target` value, return *the value in the BST that is closest to the* `target`. If there are multiple an smallest.

Example 1:

```
Input: root = [4,2,5,1,3], target = 3.714286
Output: 4
```

Example 2:

```
Input: root = [1], target = 4.428571
Output: 1
```

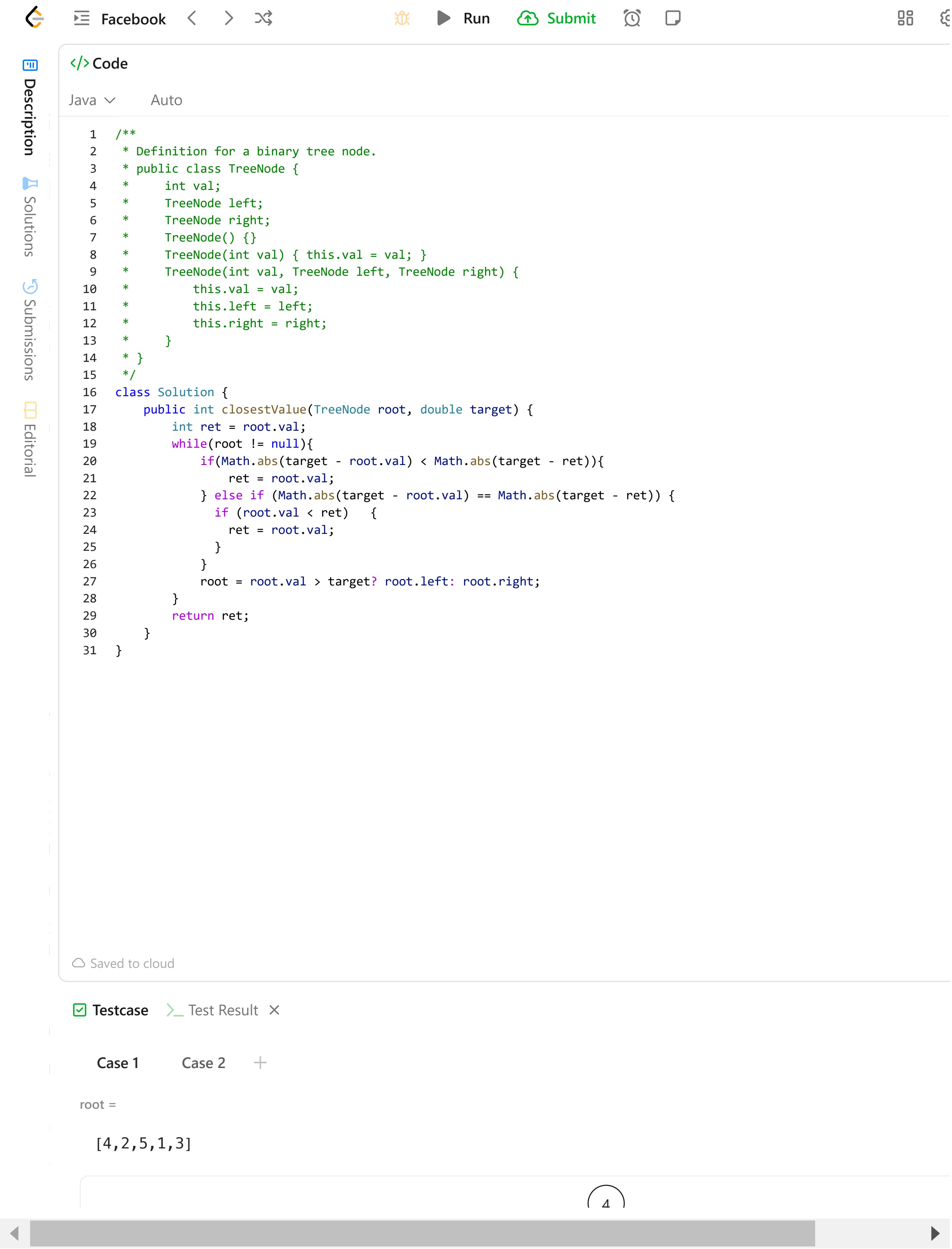
Constraints:

- The number of nodes in the tree is in the range `[1, 10^4]`.
- `0 <= Node.val <= 10^9`
- `-10^9 <= target <= 10^9`

Seen this question in a real interview before? 1/4

Yes No

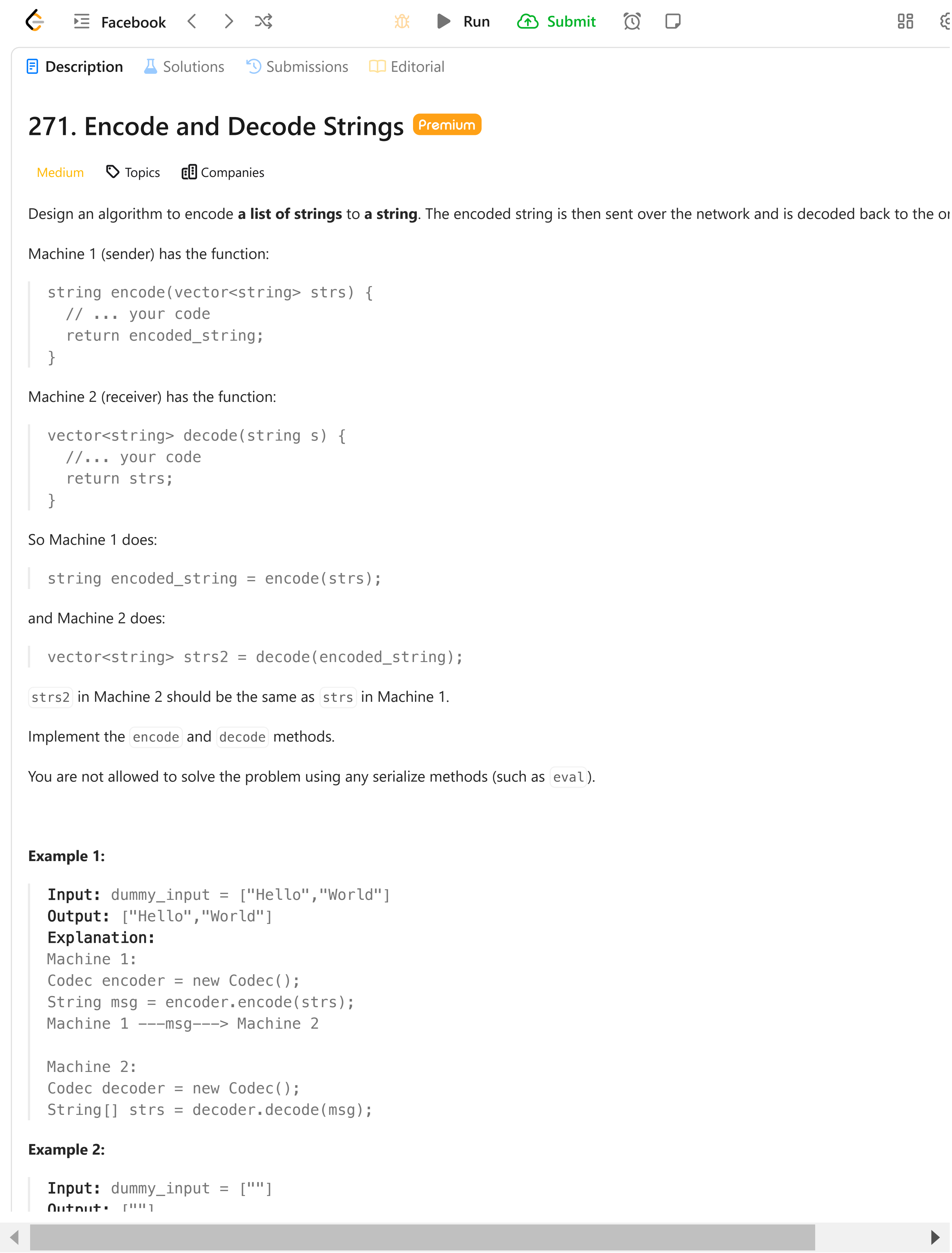
Accepted 324.7K Submissions 623.1K Acceptance Rate 52.1%



```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public int closestValue(TreeNode root, double target) {
        int ret = root.val;
        while(root != null){
            if(Math.abs(target - root.val) < Math.abs(target - ret)){
                ret = root.val;
            } else if (Math.abs(target - root.val) == Math.abs(target - ret)) {
              if (root.val < ret)   {
                ret = root.val;
              }
            }
            root = root.val > target? root.left: root.right;
        }
        return ret;
    }
}
```

root =

[4,2,5,1,3]



271. Encode and Decode Strings Premium

Medium Topics Companies

Design an algorithm to encode **a list of strings** to **a string**. The encoded string is then sent over the network and is decoded back to the or

Machine 1 (sender) has the function:

```
string encode(vector<string> strs) {
  // ... your code
  return encoded_string;
}
```

Machine 2 (receiver) has the function:

```
vector<string> decode(string s) {
  //... your code
  return strs;
}
```

So Machine 1 does:

```
string encoded_string = encode(strs);
```

and Machine 2 does:

```
vector<string> strs2 = decode(encoded_string);
```

`strs2` in Machine 2 should be the same as `strs` in Machine 1.

Implement the `encode` and `decode` methods.

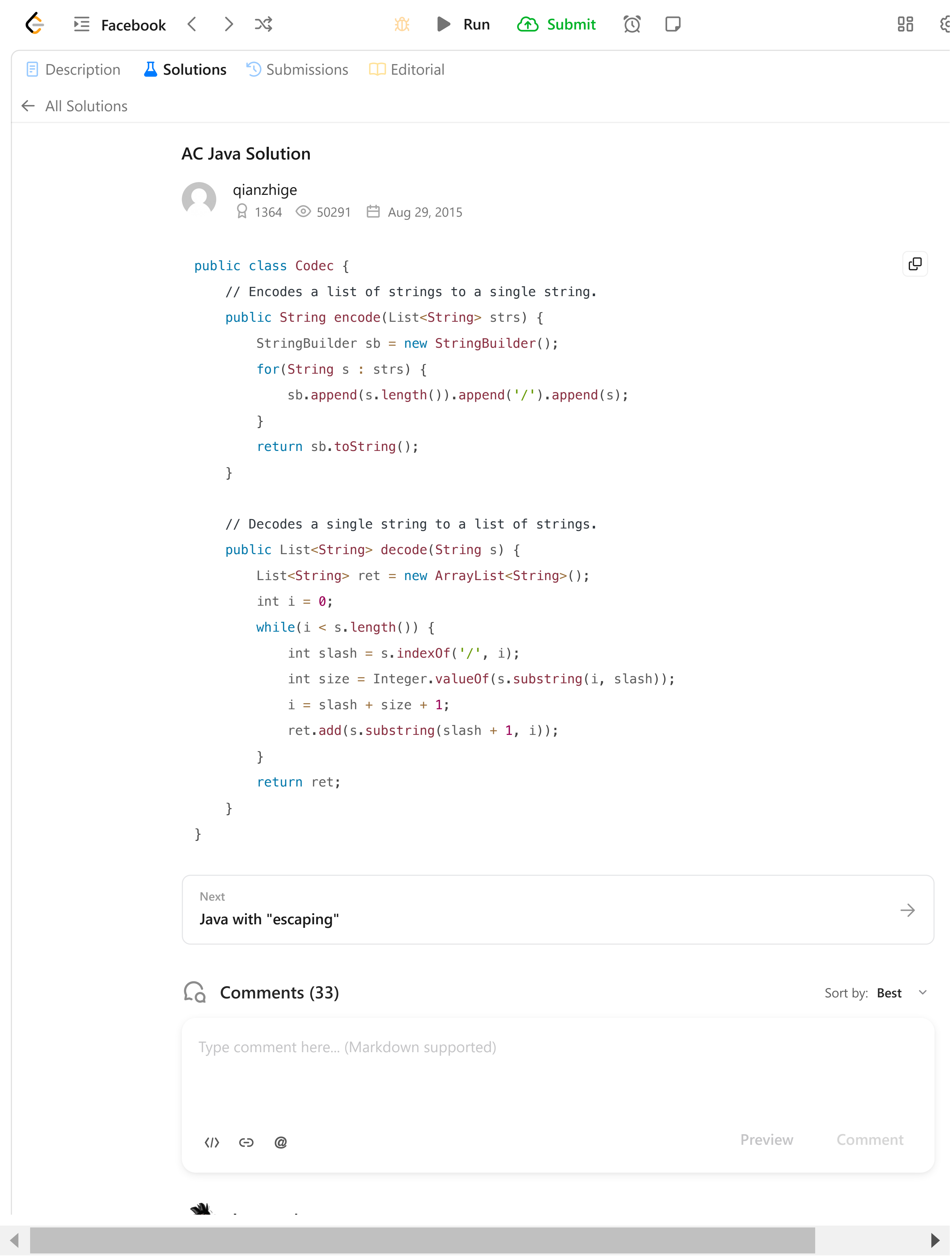
You are not allowed to solve the problem using any serialize methods (such as `eval`).

Example 1:

```
Input: dummy_input = ["Hello","World"]
Output: ["Hello","World"]
Explanation:
Machine 1:
Codec encoder = new Codec();
String msg = encoder.encode(strs);
Machine 1 ---msg---> Machine 2

Machine 2:
Codec decoder = new Codec();
String[] strs = decoder.decode(msg);
```

Example 2:

```
Input: dummy_input = [""]
Output: [""]
```



AC Java Solution

qianzhige

1364 50291 Aug 29, 2015

```java
public class Codec {
    // Encodes a list of strings to a single string.
    public String encode(List<String> strs) {
        StringBuilder sb = new StringBuilder();
        for(String s : strs) {
            sb.append(s.length()).append('/').append(s);
        }
        return sb.toString();
    }

    // Decodes a single string to a list of strings.
    public List<String> decode(String s) {
        List<String> ret = new ArrayList<String>();
        int i = 0;
        while(i < s.length()) {
            int slash = s.indexOf('/', i);
            int size = Integer.valueOf(s.substring(i, slash));
            i = slash + size + 1;
            ret.add(s.substring(slash + 1, i));
        }
        return ret;
    }
}
```

Next

Java with "escaping"

Comments (33)

Sort by: Best

Type comment here... (Markdown supported)

Facebook Run Submit

Description Solutions Submissions Editorial

277. Find the Celebrity Premium

Medium Topics Companies Hint

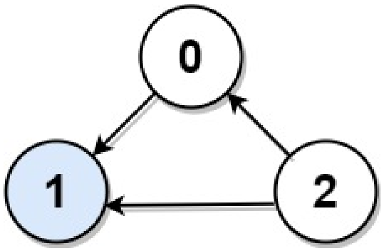
Suppose you are at a party with `n` people labeled from `0` to `n - 1` and among them, there may exist one celebrity. The definition of a cel
other `n - 1` people know the celebrity, but the celebrity does not know any of them.

Now you want to find out who the celebrity is or verify that there is not one. You are only allowed to ask questions like: "Hi, A. Do you know
information about whether A knows B. You need to find out the celebrity (or verify there is not one) by asking as few questions as possible (
sense).

You are given a helper function `bool knows(a, b)` that tells you whether `a` knows `b`. Implement a function `int findCelebrity(n)`. Ther
celebrity if they are at the party.

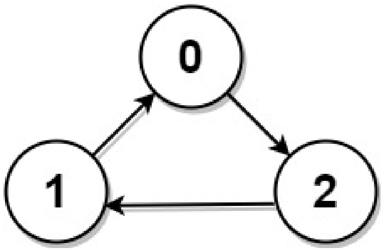
Return *the celebrity's label if there is a celebrity at the party*. If there is no celebrity, return `-1`.

Example 1:



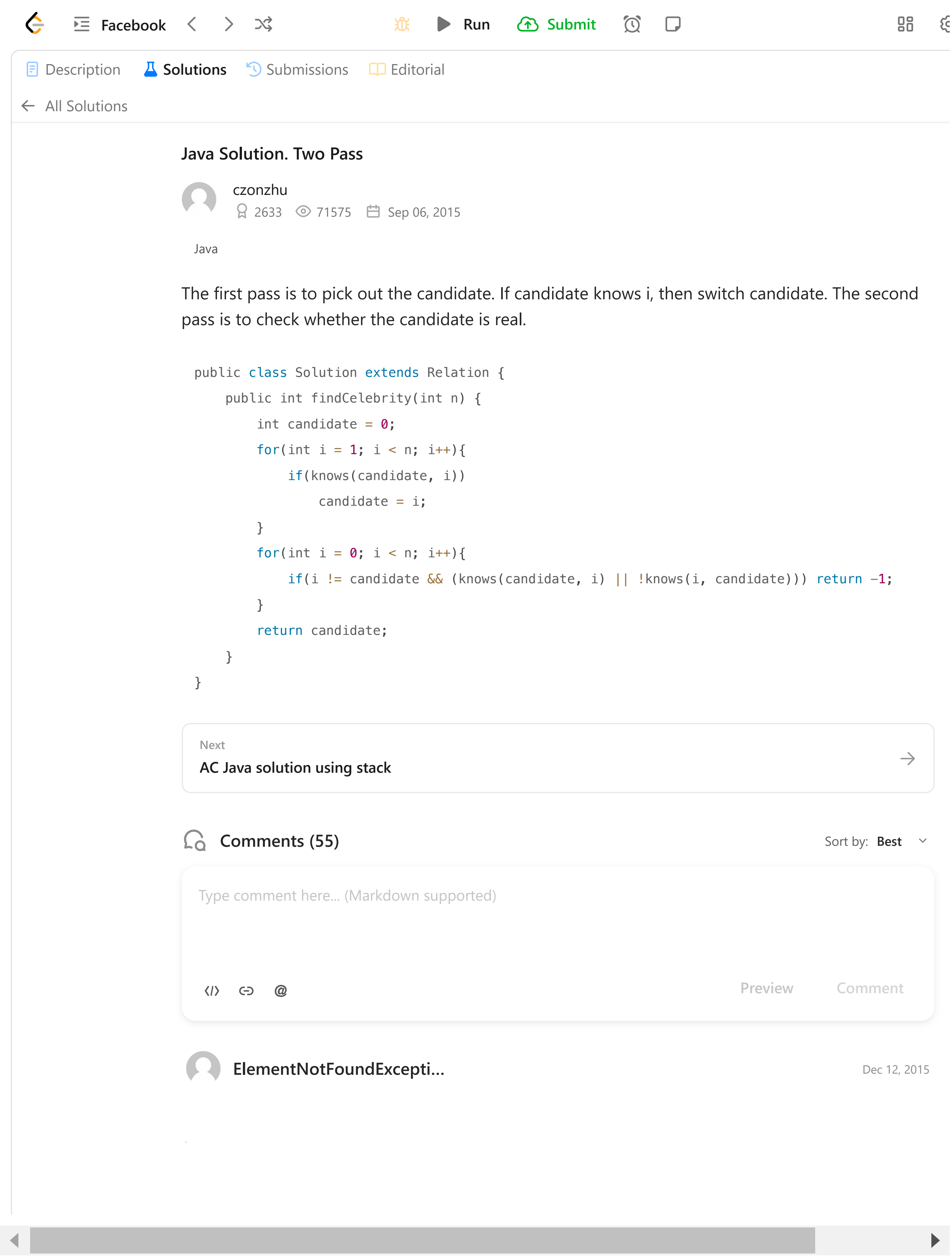
```
Input: graph = [[1,1,0],[0,1,0],[1,1,1]]
Output: 1
Explanation: There are three persons labeled with 0, 1 and 2. graph[i][j] = 1 means person i knows p
otherwise graph[i][j] = 0 means person i does not know person j. The celebrity is the person labelec
both 0 and 2 know him but 1 does not know anybody.
```

Example 2:



```
Input: graph = [[1,0,1],[1,1,0],[0,1,1]]
Output: -1
Explanation: There is no celebrity.
```

Constraints:



Java Solution. Two Pass

czonzhu
2633 71575 Sep 06, 2015

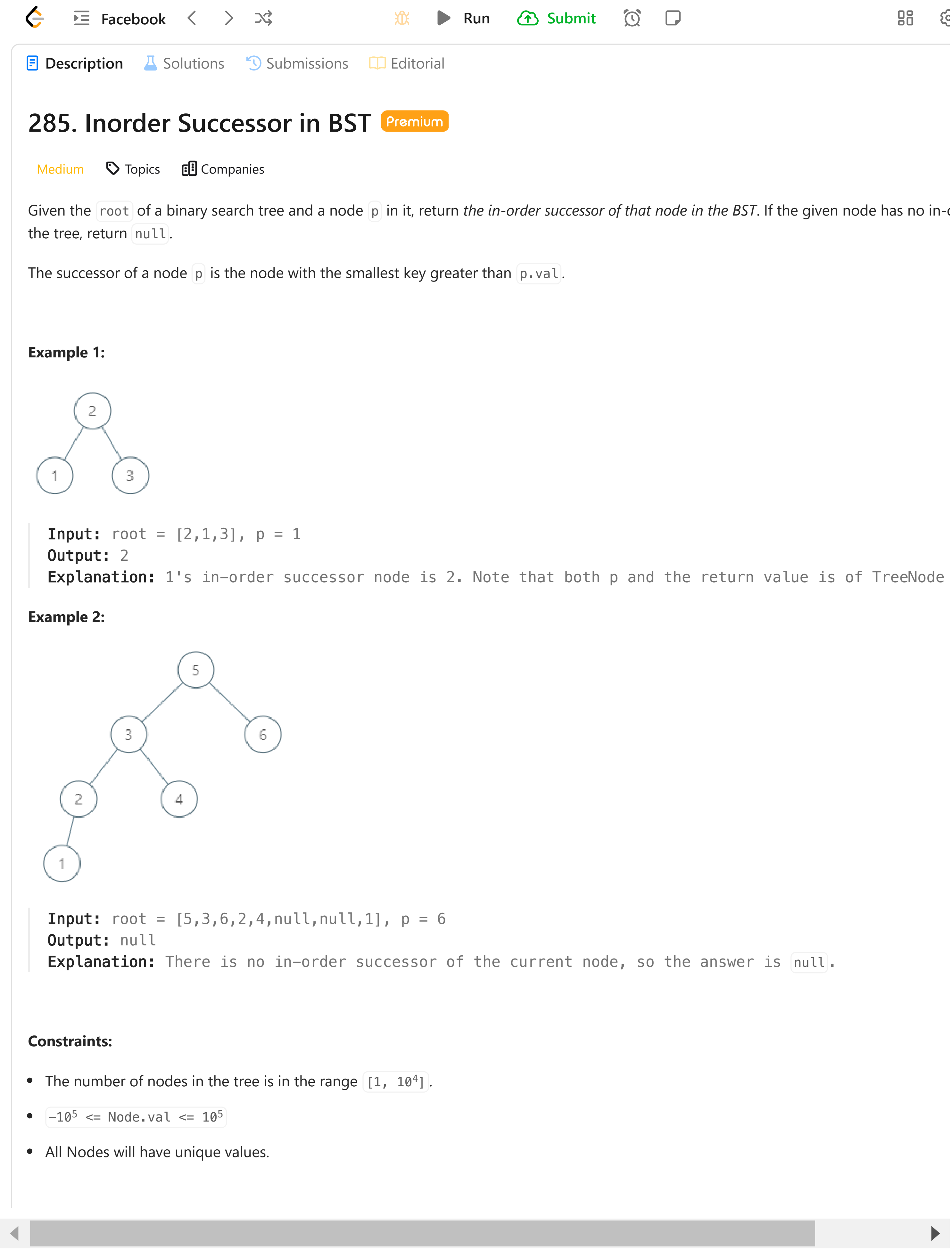
Java

The first pass is to pick out the candidate. If candidate knows i, then switch candidate. The second pass is to check whether the candidate is real.

```java
public class Solution extends Relation {
    public int findCelebrity(int n) {
        int candidate = 0;
        for(int i = 1; i < n; i++){
            if(knows(candidate, i))
                candidate = i;
        }
        for(int i = 0; i < n; i++){
            if(i != candidate && (knows(candidate, i) || !knows(i, candidate))) return -1;
        }
        return candidate;
    }
}
```

Comments (55)

ElementNotFoundExcepti... Dec 12, 2015



285. Inorder Successor in BST

Given the `root` of a binary search tree and a node `p` in it, return *the in-order successor of that node in the BST*. If the given node has no in-(
the tree, return `null`.

The successor of a node `p` is the node with the smallest key greater than `p.val`.

Example 1:

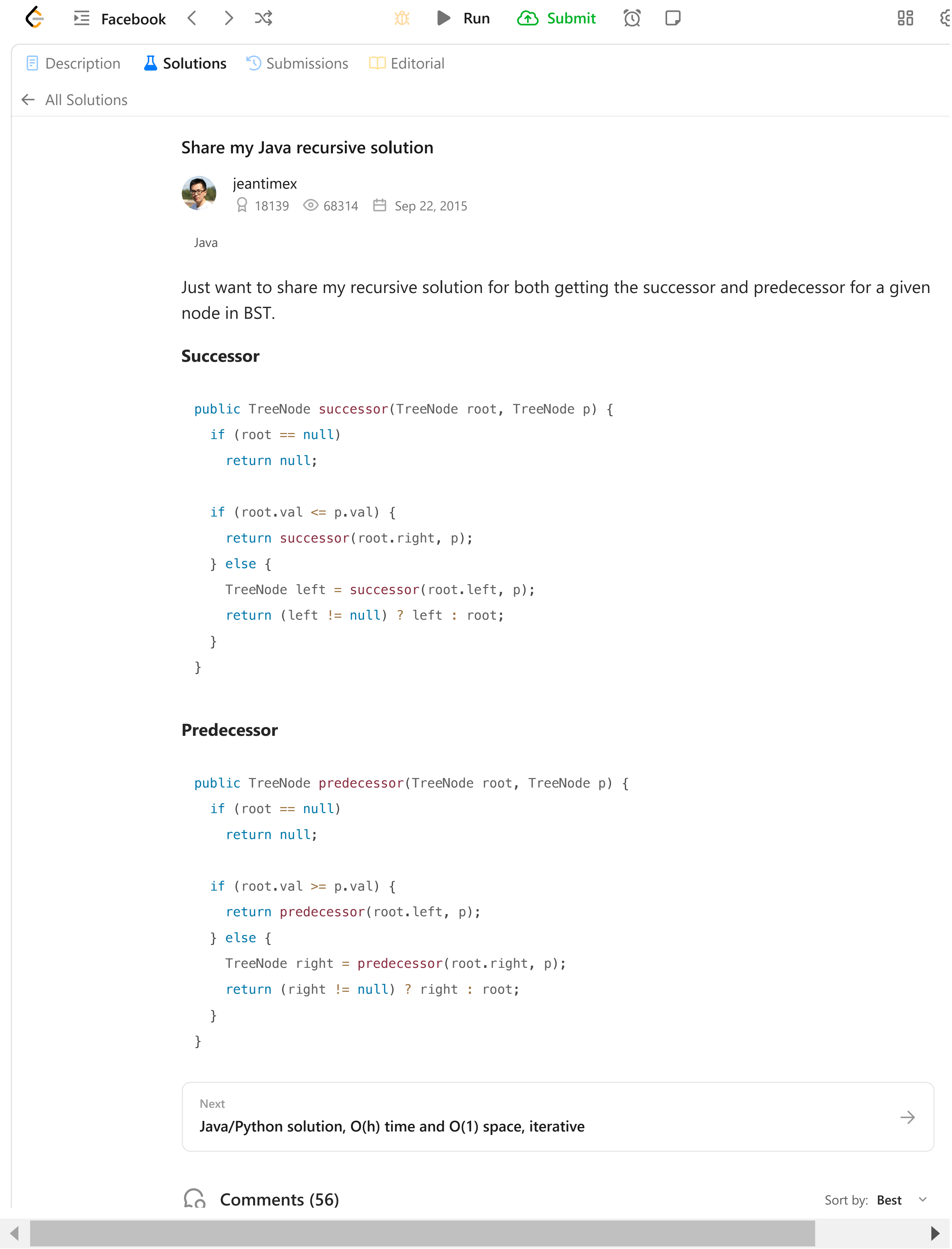
```
Input: root = [2,1,3], p = 1
Output: 2
Explanation: 1's in-order successor node is 2. Note that both p and the return value is of TreeNode
```

Example 2:

```
Input: root = [5,3,6,2,4,null,null,1], p = 6
Output: null
Explanation: There is no in-order successor of the current node, so the answer is null.
```

Constraints:

- The number of nodes in the tree is in the range `[1, 10^4]`.
- `-10^5 <= Node.val <= 10^5`
- All Nodes will have unique values.



Share my Java recursive solution

jeantimex
18139 68314 Sep 22, 2015

Java

Just want to share my recursive solution for both getting the successor and predecessor for a given node in BST.

Successor

```
public TreeNode successor(TreeNode root, TreeNode p) {
  if (root == null)
    return null;

  if (root.val <= p.val) {
    return successor(root.right, p);
  } else {
    TreeNode left = successor(root.left, p);
    return (left != null) ? left : root;
  }
}
```

Predecessor

```
public TreeNode predecessor(TreeNode root, TreeNode p) {
  if (root == null)
    return null;

  if (root.val >= p.val) {
    return predecessor(root.left, p);
  } else {
    TreeNode right = predecessor(root.right, p);
    return (right != null) ? right : root;
  }
}
```

Next
Java/Python solution, O(h) time and O(1) space, iterative

Comments (56)

Sort by: Best

Facebook Run Submit

Description Solutions Submissions Editorial

286. Walls and Gates Premium

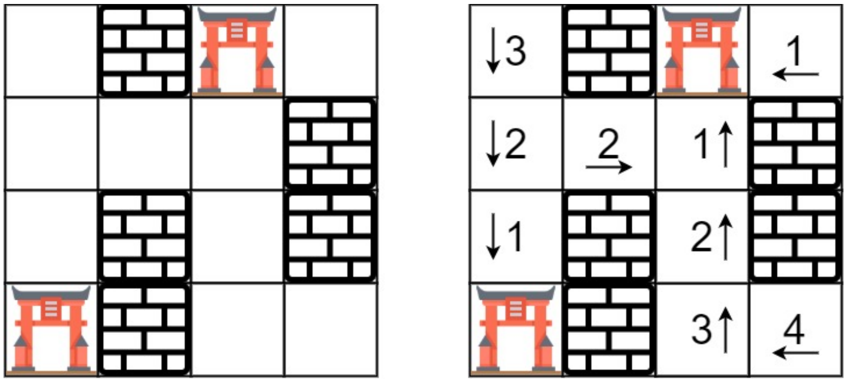
Medium Topics Companies

You are given an `m x n` grid `rooms` initialized with these three possible values.

- `-1` A wall or an obstacle.
- `0` A gate.
- `INF` Infinity means an empty room. We use the value $2^{31} - 1 = 2147483647$ to represent `INF` as you may assume that the distance to `2147483647`.

Fill each empty room with the distance to *its nearest gate*. If it is impossible to reach a gate, it should be filled with `INF`.

Example 1:



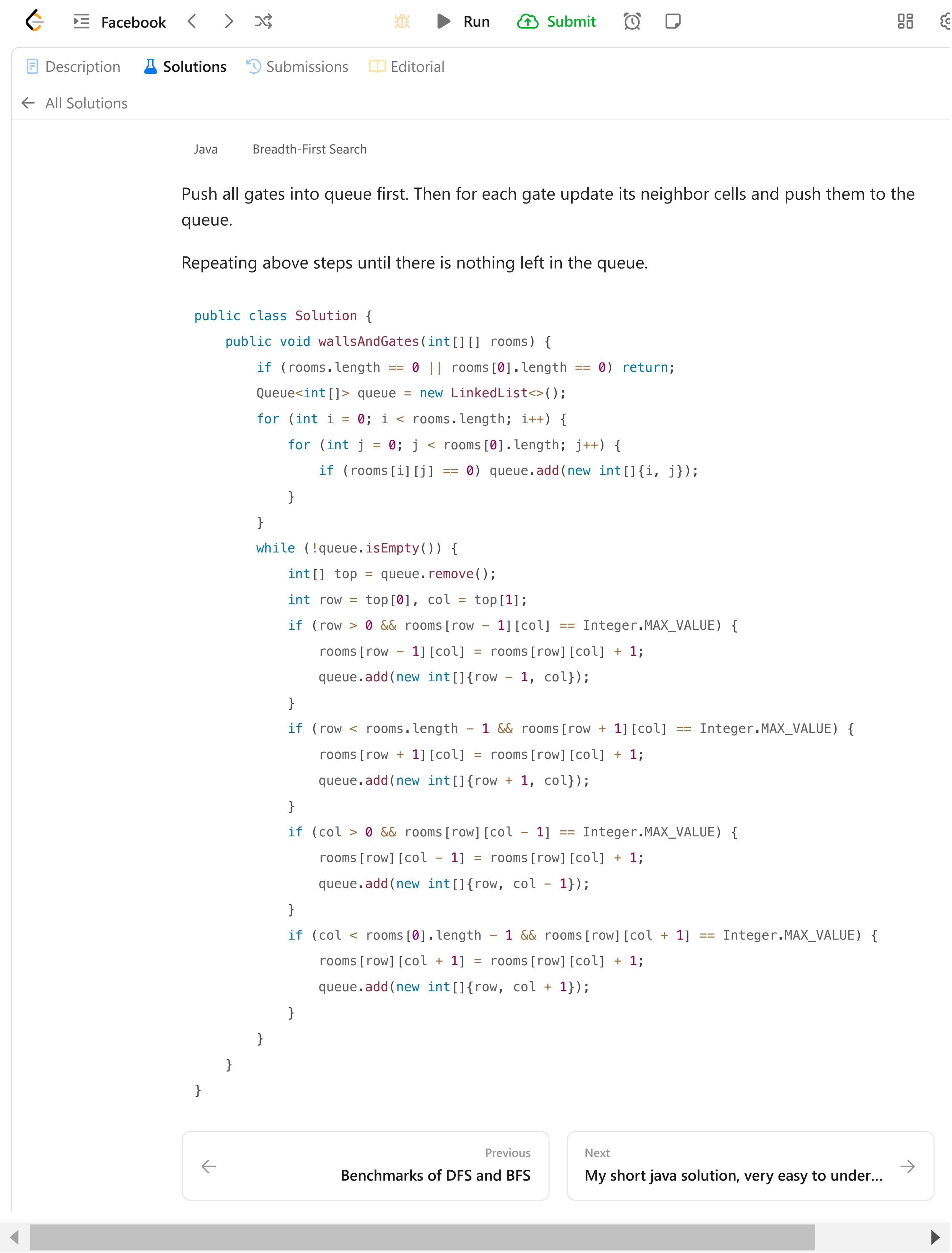
```
Input: rooms = [[2147483647,-1,0,2147483647],[2147483647,2147483647,2147483647,-1],[2147483647,-1,21
[0,-1,2147483647,2147483647]]
Output: [[3,-1,0,1],[2,2,1,-1],[1,-1,2,-1],[0,-1,3,4]]
```

Example 2:

```
Input: rooms = [[-1]]
Output: [[-1]]
```

Constraints:

- `m == rooms.length`
- `n == rooms[i].length`
- `1 <= m, n <= 250`
- `rooms[i][j]` is `-1`, `0`, or $2^{31} - 1$.

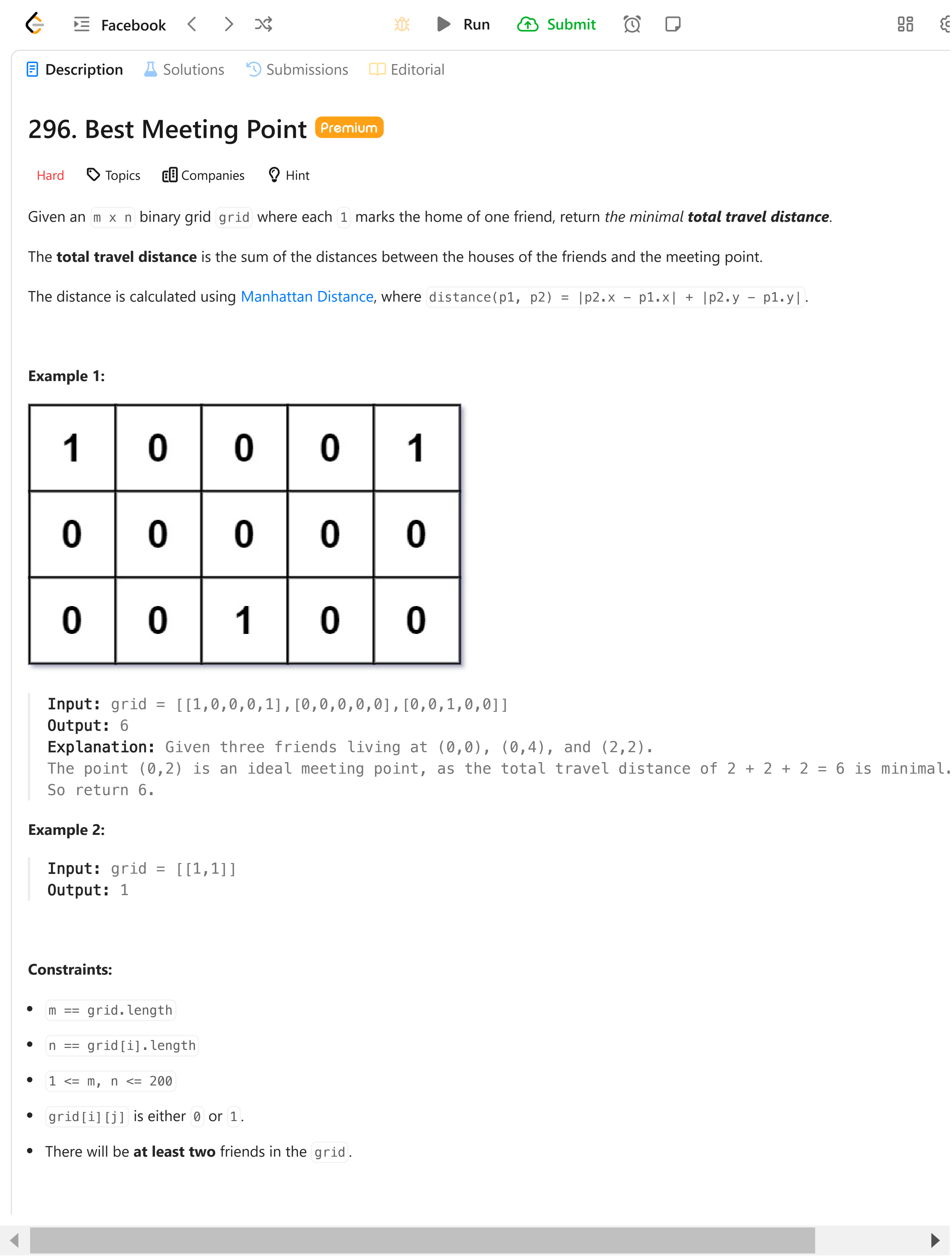


Push all gates into queue first. Then for each gate update its neighbor cells and push them to the queue.

Repeating above steps until there is nothing left in the queue.

```java
public class Solution {
    public void wallsAndGates(int[][] rooms) {
        if (rooms.length == 0 || rooms[0].length == 0) return;
        Queue<int[]> queue = new LinkedList<>();
        for (int i = 0; i < rooms.length; i++) {
            for (int j = 0; j < rooms[0].length; j++) {
                if (rooms[i][j] == 0) queue.add(new int[]{i, j});
            }
        }
        while (!queue.isEmpty()) {
            int[] top = queue.remove();
            int row = top[0], col = top[1];
            if (row > 0 && rooms[row - 1][col] == Integer.MAX_VALUE) {
                rooms[row - 1][col] = rooms[row][col] + 1;
                queue.add(new int[]{row - 1, col});
            }
            if (row < rooms.length - 1 && rooms[row + 1][col] == Integer.MAX_VALUE) {
                rooms[row + 1][col] = rooms[row][col] + 1;
                queue.add(new int[]{row + 1, col});
            }
            if (col > 0 && rooms[row][col - 1] == Integer.MAX_VALUE) {
                rooms[row][col - 1] = rooms[row][col] + 1;
                queue.add(new int[]{row, col - 1});
            }
            if (col < rooms[0].length - 1 && rooms[row][col + 1] == Integer.MAX_VALUE) {
                rooms[row][col + 1] = rooms[row][col] + 1;
                queue.add(new int[]{row, col + 1});
            }
        }
    }
}
```

Previous
Benchmarks of DFS and BFS
Next
My short java solution, very easy to under...



296. Best Meeting Point Premium

Hard Topics Companies Hint

Given an `m x n` binary grid `grid` where each `1` marks the home of one friend, return *the minimal* ***total travel distance***.

The **total travel distance** is the sum of the distances between the houses of the friends and the meeting point.

The distance is calculated using Manhattan Distance, where `distance(p1, p2) = |p2.x - p1.x| + |p2.y - p1.y|`.

Example 1:

1	0	0	0	1
0	0	0	0	0
0	0	1	0	0

```
Input: grid = [[1,0,0,0,1],[0,0,0,0,0],[0,0,1,0,0]]
Output: 6
Explanation: Given three friends living at (0,0), (0,4), and (2,2).
The point (0,2) is an ideal meeting point, as the total travel distance of 2 + 2 + 2 = 6 is minimal.
So return 6.
```

Example 2:

```
Input: grid = [[1,1]]
Output: 1
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 200`
- `grid[i][j]` is either `0` or `1`.
- There will be **at least two** friends in the `grid`.

14ms java solution

larrywang2014

1206 31871 Oct 21, 2015

```
public int minTotalDistance(int[][] grid) {
    int m = grid.length;
    int n = grid[0].length;

    List<Integer> I = new ArrayList<>(m);
    List<Integer> J = new ArrayList<>(n);

    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            if(grid[i][j] == 1){
                I.add(i);
                J.add(j);
            }
        }
    }

    return getMin(I) + getMin(J);
}

private int getMin(List<Integer> list){
    int ret = 0;

    Collections.sort(list);

    int i = 0;
    int j = list.size() - 1;
    while(i < j){
        ret += list.get(j--) - list.get(i++);
    }

    return ret;
}
```

Next

Java 2ms/Python 40ms two pointers solution no median no sort with explanation

298. Binary Tree Longest Consecutive Sequence Premium

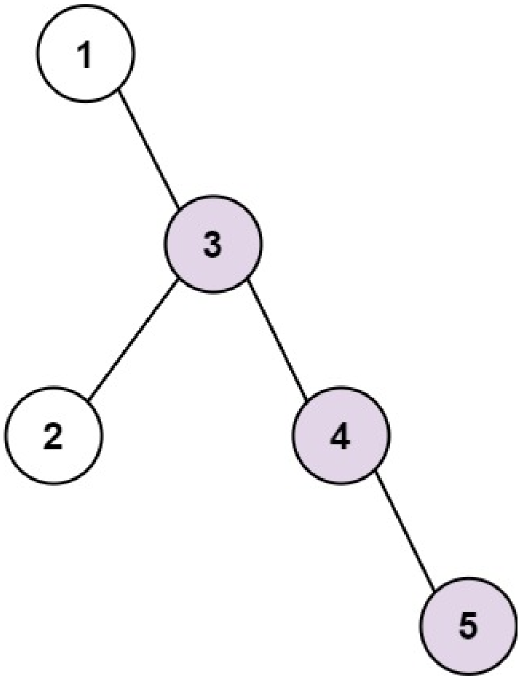
Medium Topics Companies

Given the `root` of a binary tree, return *the length of the longest* ***consecutive sequence path***.

A **consecutive sequence path** is a path where the values **increase by one** along the path.

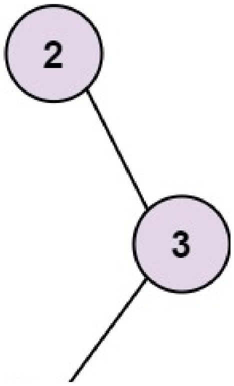
Note that the path can start **at any node** in the tree, and you cannot go from a node to its parent in the path.

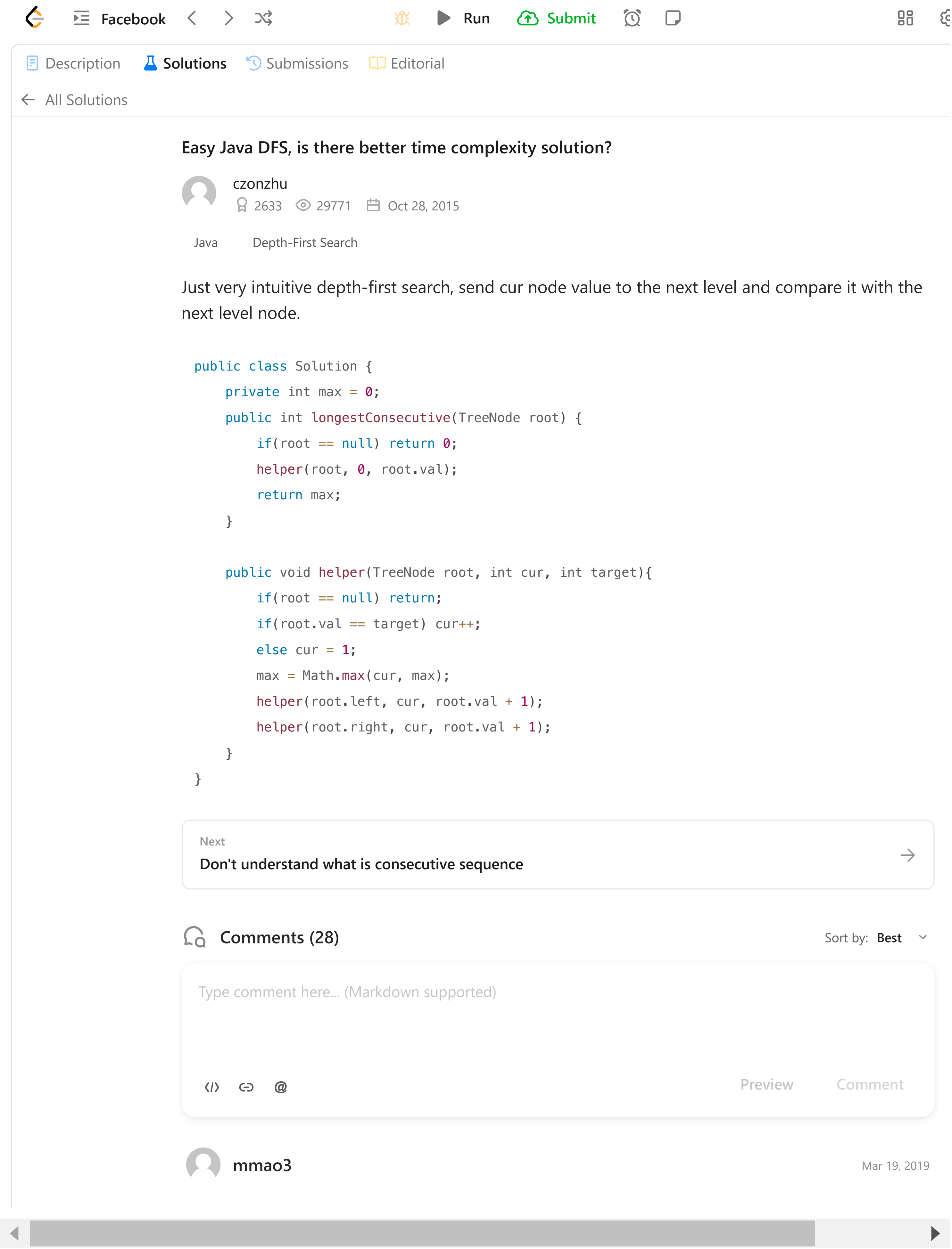
Example 1:



```
Input: root = [1,null,3,2,4,null,null,null,5]
Output: 3
Explanation: Longest consecutive sequence path is 3-4-5, so return 3.
```

Example 2:





Easy Java DFS, is there better time complexity solution?

czonzhu
2633 29771 Oct 28, 2015

Java Depth-First Search

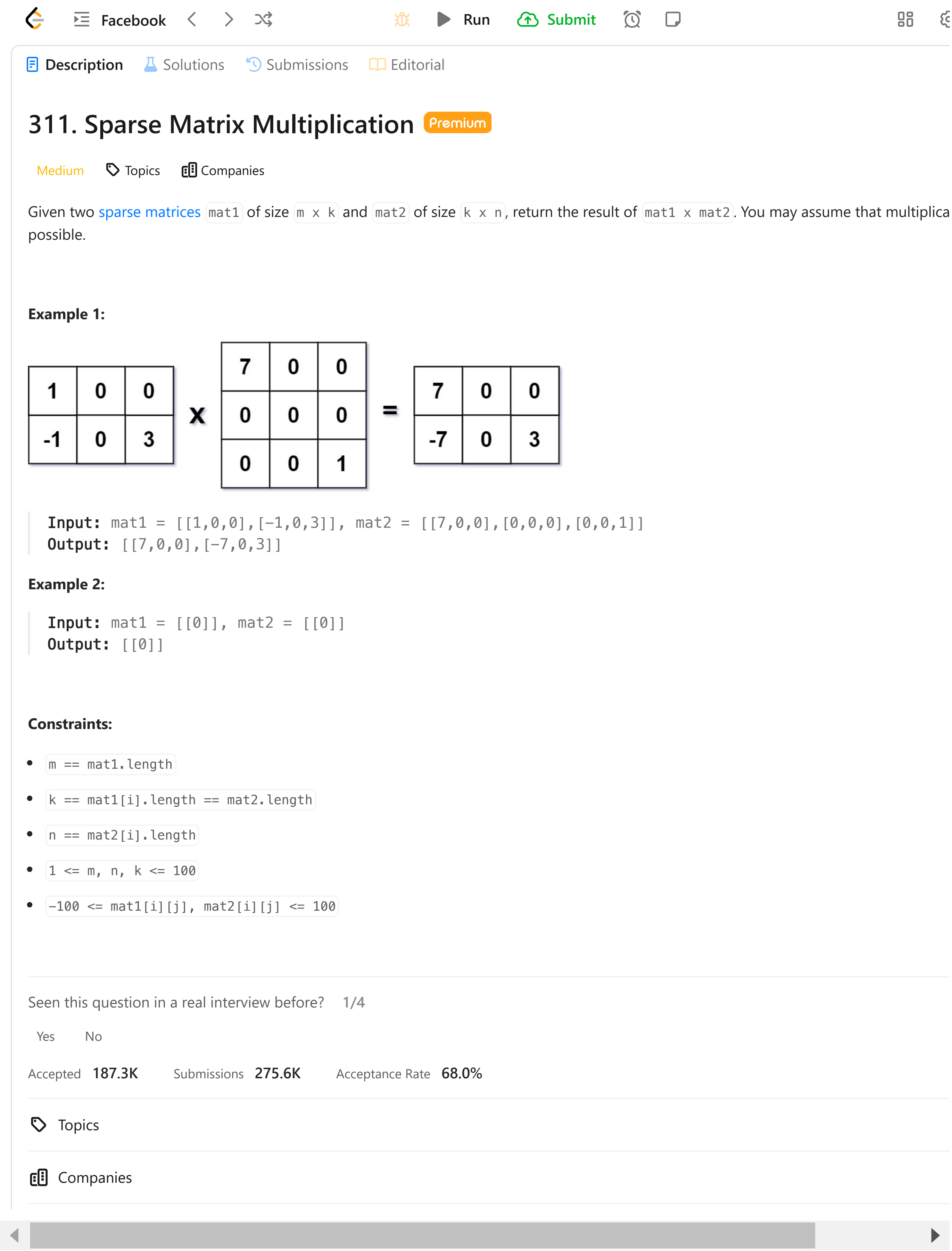
Just very intuitive depth-first search, send cur node value to the next level and compare it with the next level node.

```
public class Solution {
    private int max = 0;
    public int longestConsecutive(TreeNode root) {
        if(root == null) return 0;
        helper(root, 0, root.val);
        return max;
    }

    public void helper(TreeNode root, int cur, int target){
        if(root == null) return;
        if(root.val == target) cur++;
        else cur = 1;
        max = Math.max(cur, max);
        helper(root.left, cur, root.val + 1);
        helper(root.right, cur, root.val + 1);
    }
}
```

Next
Don't understand what is consecutive sequence

Comments (28)

mmao3 Mar 19, 2019



311. Sparse Matrix Multiplication Premium

Medium Topics Companies

Given two sparse matrices `mat1` of size `m x k` and `mat2` of size `k x n`, return the result of `mat1 x mat2`. You may assume that multiplica possible.

Example 1:

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 3 \end{bmatrix} \times \begin{bmatrix} 7 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 0 & 0 \\ -7 & 0 & 3 \end{bmatrix}$$

```
Input: mat1 = [[1,0,0],[-1,0,3]], mat2 = [[7,0,0],[0,0,0],[0,0,1]]
Output: [[7,0,0],[-7,0,3]]
```

Example 2:

```
Input: mat1 = [[0]], mat2 = [[0]]
Output: [[0]]
```

Constraints:

- `m == mat1.length`
- `k == mat1[i].length == mat2.length`
- `n == mat2[i].length`
- `1 <= m, n, k <= 100`
- `-100 <= mat1[i][j], mat2[i][j] <= 100`

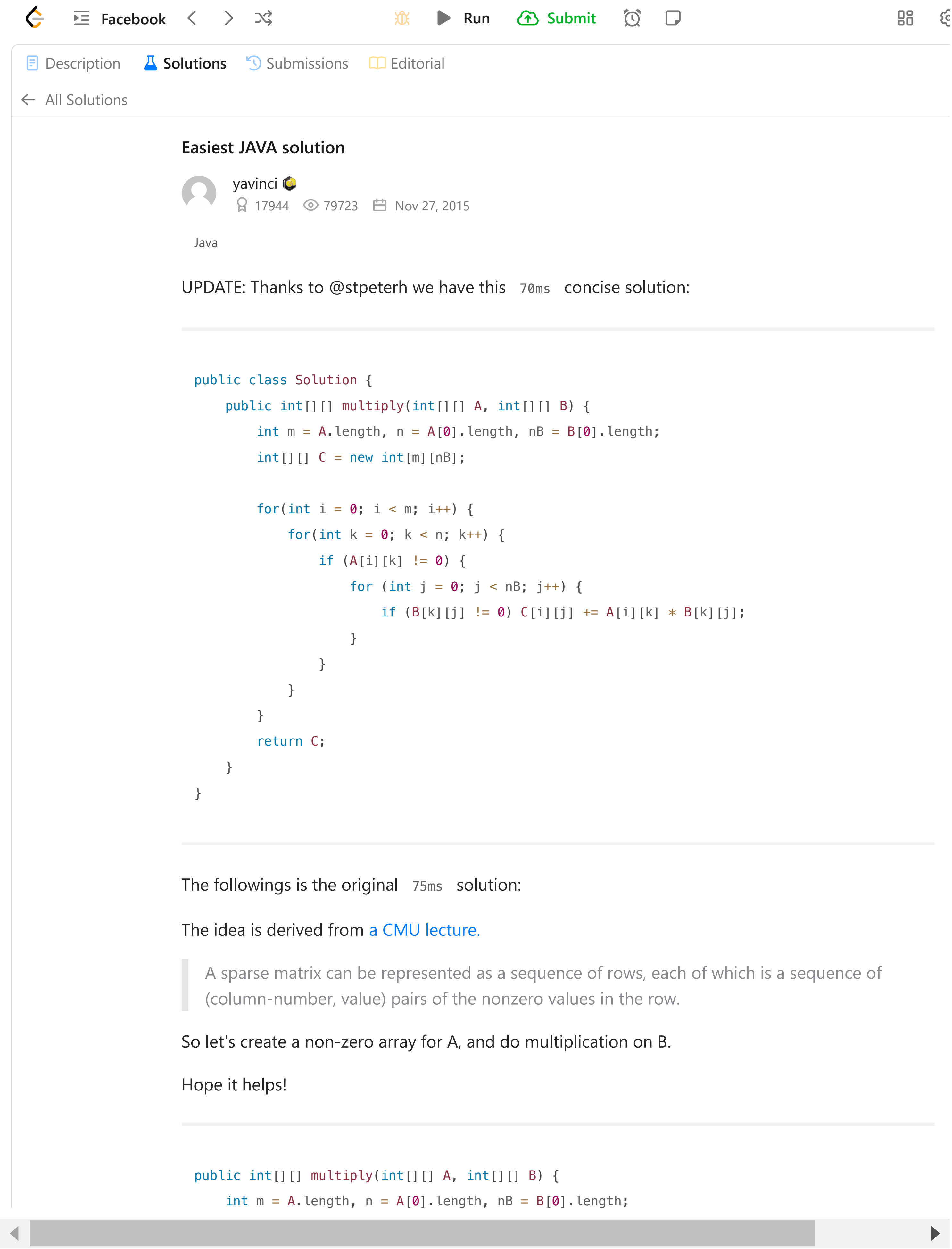
Seen this question in a real interview before? 1/4

Yes No

Accepted 187.3K Submissions 275.6K Acceptance Rate 68.0%

Topics

Companies



Easiest JAVA solution

yavinci

17944 79723 Nov 27, 2015

Java

UPDATE: Thanks to @stpeterh we have this `70ms` concise solution:

```
public class Solution {
    public int[][] multiply(int[][] A, int[][] B) {
        int m = A.length, n = A[0].length, nB = B[0].length;
        int[][] C = new int[m][nB];

        for(int i = 0; i < m; i++) {
            for(int k = 0; k < n; k++) {
                if (A[i][k] != 0) {
                    for (int j = 0; j < nB; j++) {
                        if (B[k][j] != 0) C[i][j] += A[i][k] * B[k][j];
                    }
                }
            }
        }
        return C;
    }
}
```

The followings is the original `75ms` solution:

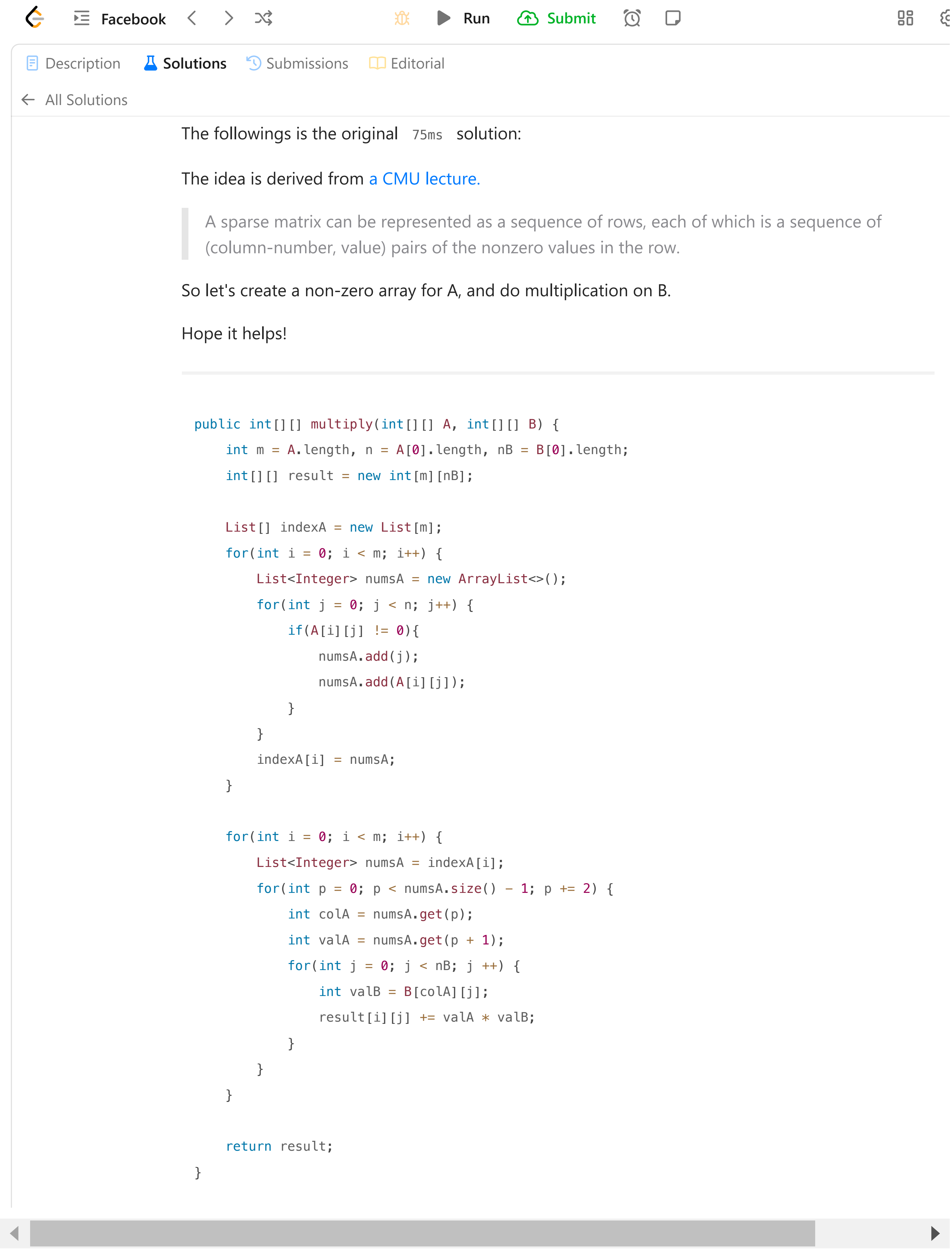
The idea is derived from a CMU lecture.

> A sparse matrix can be represented as a sequence of rows, each of which is a sequence of (column-number, value) pairs of the nonzero values in the row.

So let's create a non-zero array for A, and do multiplication on B.

Hope it helps!

```
public int[][] multiply(int[][] A, int[][] B) {
    int m = A.length, n = A[0].length, nB = B[0].length;
```



The followings is the original `75ms` solution:

The idea is derived from a CMU lecture.

> A sparse matrix can be represented as a sequence of rows, each of which is a sequence of (column-number, value) pairs of the nonzero values in the row.

So let's create a non-zero array for A, and do multiplication on B.

Hope it helps!

```java
public int[][] multiply(int[][] A, int[][] B) {
    int m = A.length, n = A[0].length, nB = B[0].length;
    int[][] result = new int[m][nB];

    List[] indexA = new List[m];
    for(int i = 0; i < m; i++) {
        List<Integer> numsA = new ArrayList<>();
        for(int j = 0; j < n; j++) {
            if(A[i][j] != 0){
                numsA.add(j);
                numsA.add(A[i][j]);
            }
        }
        indexA[i] = numsA;
    }

    for(int i = 0; i < m; i++) {
        List<Integer> numsA = indexA[i];
        for(int p = 0; p < numsA.size() - 1; p += 2) {
            int colA = numsA.get(p);
            int valA = numsA.get(p + 1);
            for(int j = 0; j < nB; j ++) {
                int valB = B[colA][j];
                result[i][j] += valA * valB;
            }
        }
    }

    return result;
}
```

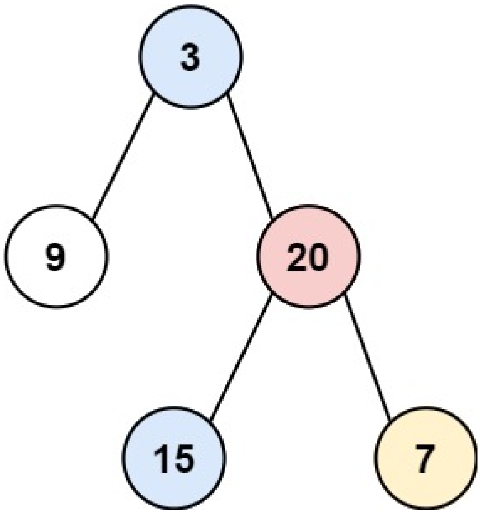
314. Binary Tree Vertical Order Traversal Premium

Medium Topics Companies Hint

Given the `root` of a binary tree, return ***the vertical order traversal** of its nodes' values*. (i.e., from top to bottom, column by column).

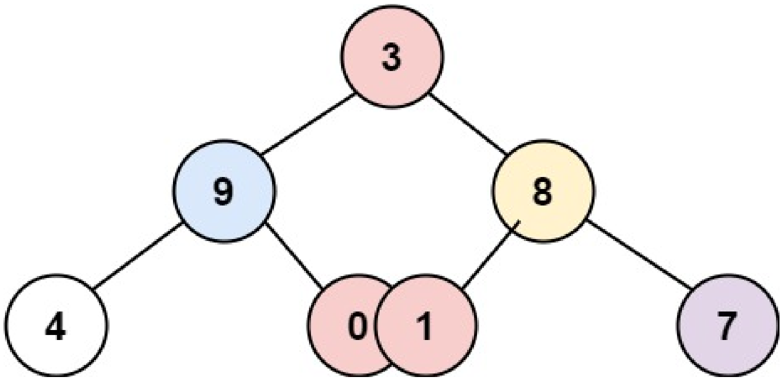
If two nodes are in the same row and column, the order should be from **left to right**.

Example 1:



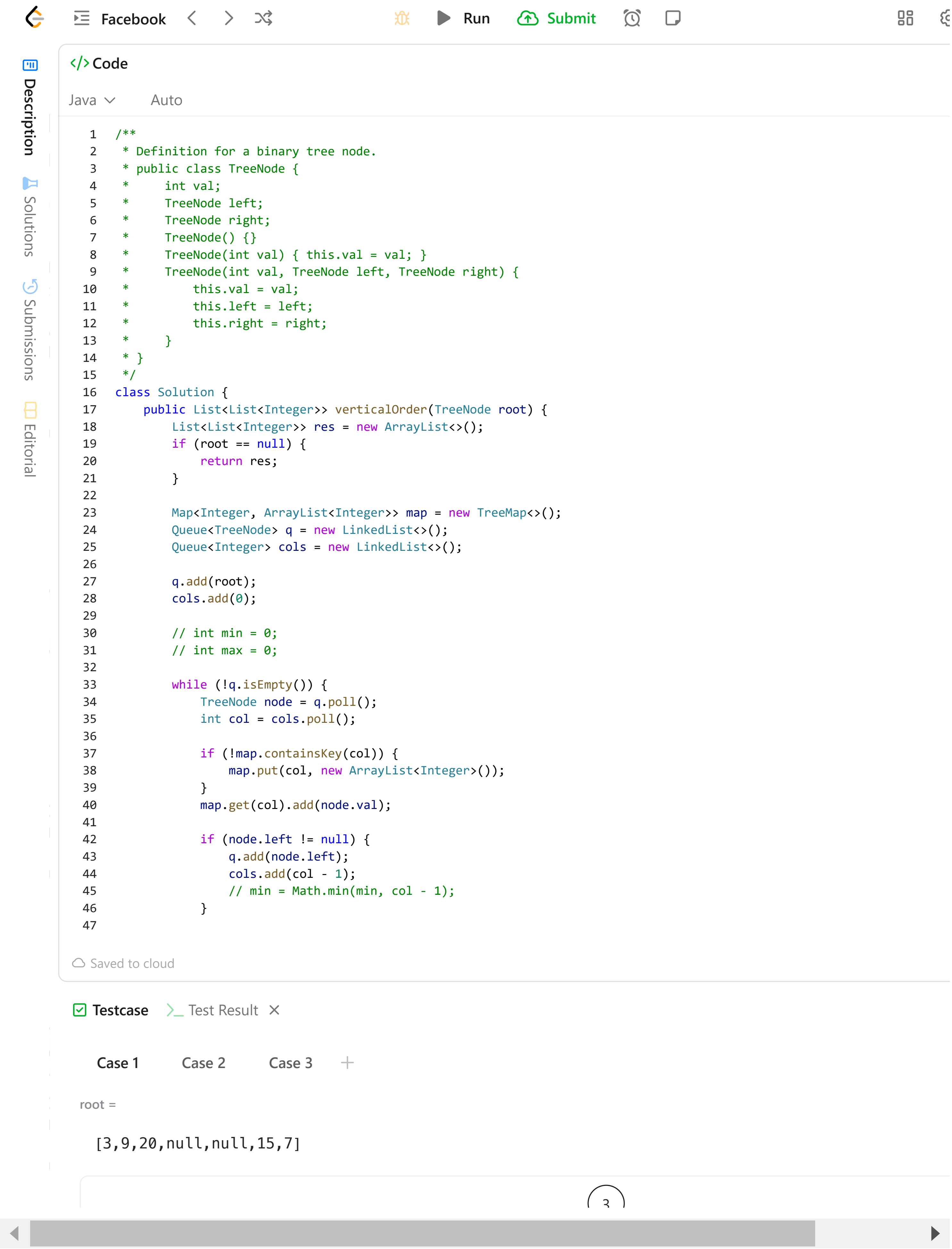
```
Input: root = [3,9,20,null,null,15,7]
Output: [[9],[3,15],[20],[7]]
```

Example 2:



```
Input: root = [3,9,8,4,0,1,7]
Output: [[4],[9],[3,0,1],[8],[7]]
```

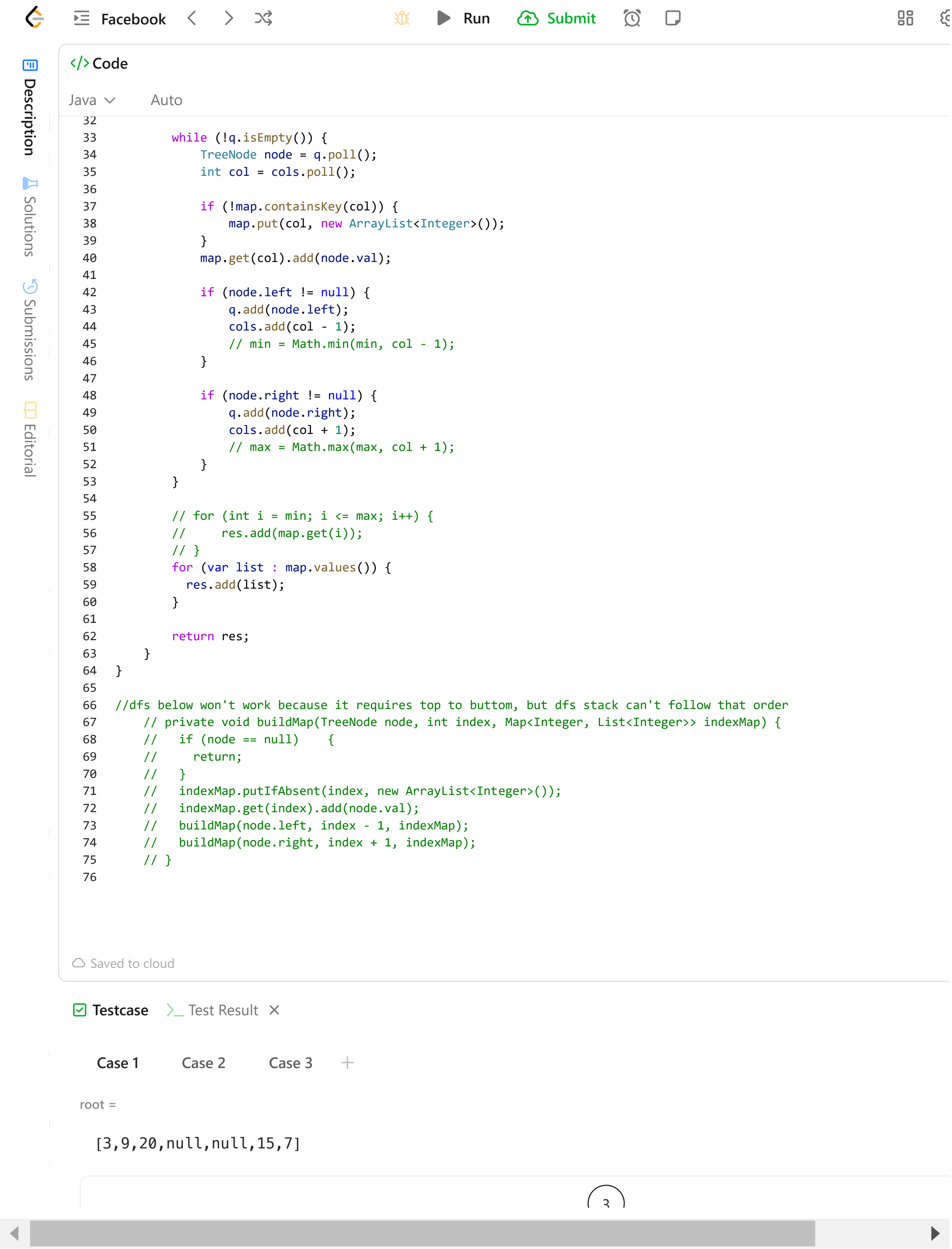
Example 3:



```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public List<List<Integer>> verticalOrder(TreeNode root) {
        List<List<Integer>> res = new ArrayList<>();
        if (root == null) {
            return res;
        }

        Map<Integer, ArrayList<Integer>> map = new TreeMap<>();
        Queue<TreeNode> q = new LinkedList<>();
        Queue<Integer> cols = new LinkedList<>();

        q.add(root);
        cols.add(0);

        // int min = 0;
        // int max = 0;

        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            int col = cols.poll();

            if (!map.containsKey(col)) {
                map.put(col, new ArrayList<Integer>());
            }
            map.get(col).add(node.val);

            if (node.left != null) {
                q.add(node.left);
                cols.add(col - 1);
                // min = Math.min(min, col - 1);
            }

```

root =

[3,9,20,null,null,15,7]



```java

        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            int col = cols.poll();

            if (!map.containsKey(col)) {
                map.put(col, new ArrayList<Integer>());
            }
            map.get(col).add(node.val);

            if (node.left != null) {
                q.add(node.left);
                cols.add(col - 1);
                // min = Math.min(min, col - 1);
            }

            if (node.right != null) {
                q.add(node.right);
                cols.add(col + 1);
                // max = Math.max(max, col + 1);
            }
        }

        // for (int i = min; i <= max; i++) {
        //     res.add(map.get(i));
        // }
        for (var list : map.values()) {
          res.add(list);
        }

        return res;
    }
}

//dfs below won't work because it requires top to buttom, but dfs stack can't follow that order
    // private void buildMap(TreeNode node, int index, Map<Integer, List<Integer>> indexMap) {
    //   if (node == null)    {
    //     return;
    //   }
    //   indexMap.putIfAbsent(index, new ArrayList<Integer>());
    //   indexMap.get(index).add(node.val);
    //   buildMap(node.left, index - 1, indexMap);
    //   buildMap(node.right, index + 1, indexMap);
    // }

```

root =

[3,9,20,null,null,15,7]

317. Shortest Distance from All Buildings Premium

Hard Topics Companies

You are given an `m x n` grid `grid` of values `0`, `1`, or `2`, where:

- each `0` marks **an empty land** that you can pass by freely,
- each `1` marks **a building** that you cannot pass through, and
- each `2` marks **an obstacle** that you cannot pass through.

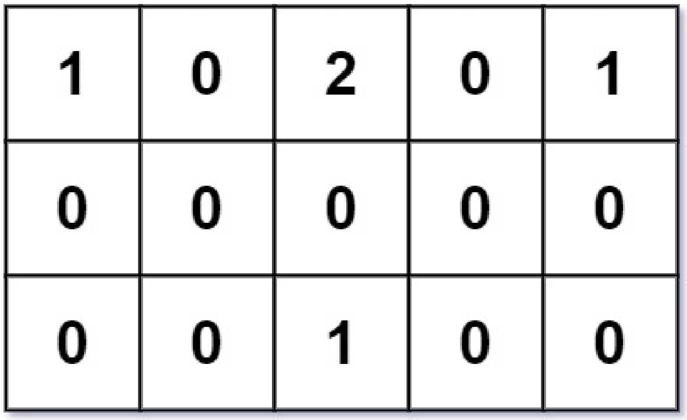
You want to build a house on an empty land that reaches all buildings in the **shortest total travel** distance. You can only move up, down, le

Return *the* ***shortest travel distance*** *for such a house*. If it is not possible to build such a house according to the above rules, return `-1`.

The **total travel distance** is the sum of the distances between the houses of the friends and the meeting point.

The distance is calculated using Manhattan Distance, where `distance(p1, p2) = |p2.x - p1.x| + |p2.y - p1.y|`.

Example 1:



1	0	2	0	1
0	0	0	0	0
0	0	1	0	0

```
Input: grid = [[1,0,2,0,1],[0,0,0,0,0],[0,0,1,0,0]]
Output: 7
Explanation: Given three buildings at (0,0), (0,4), (2,2), and an obstacle at (0,2).
The point (1,2) is an ideal empty land to build a house, as the total travel distance of 3+3+1=7 is
So return 7.
```

Example 2:

```
Input: grid = [[1,0]]
Output: 1
```

Example 3:

```
Input: grid = [[1]]
Output: -1
```

Facebook Run Submit

Description Solutions Submissions Editorial

All Solutions

shuoshankou

1065 69836 Dec 18, 2015

Java

Inspired by previous solution.
The main idea is the following:

Traverse the matrix. For each building, use BFS to compute the shortest distance from each '0' to this building. After we do this for all the buildings, we can get the sum of shortest distance from every '0' to all reachable buildings. This value is stored in 'distance[][]'. For example, if grid[2][2] == 0, distance[2][2] is the sum of shortest distance from this block to all reachable buildings.
Time complexity: O(number of 1)*O(number of 0) ~ O(m^2*n^2)

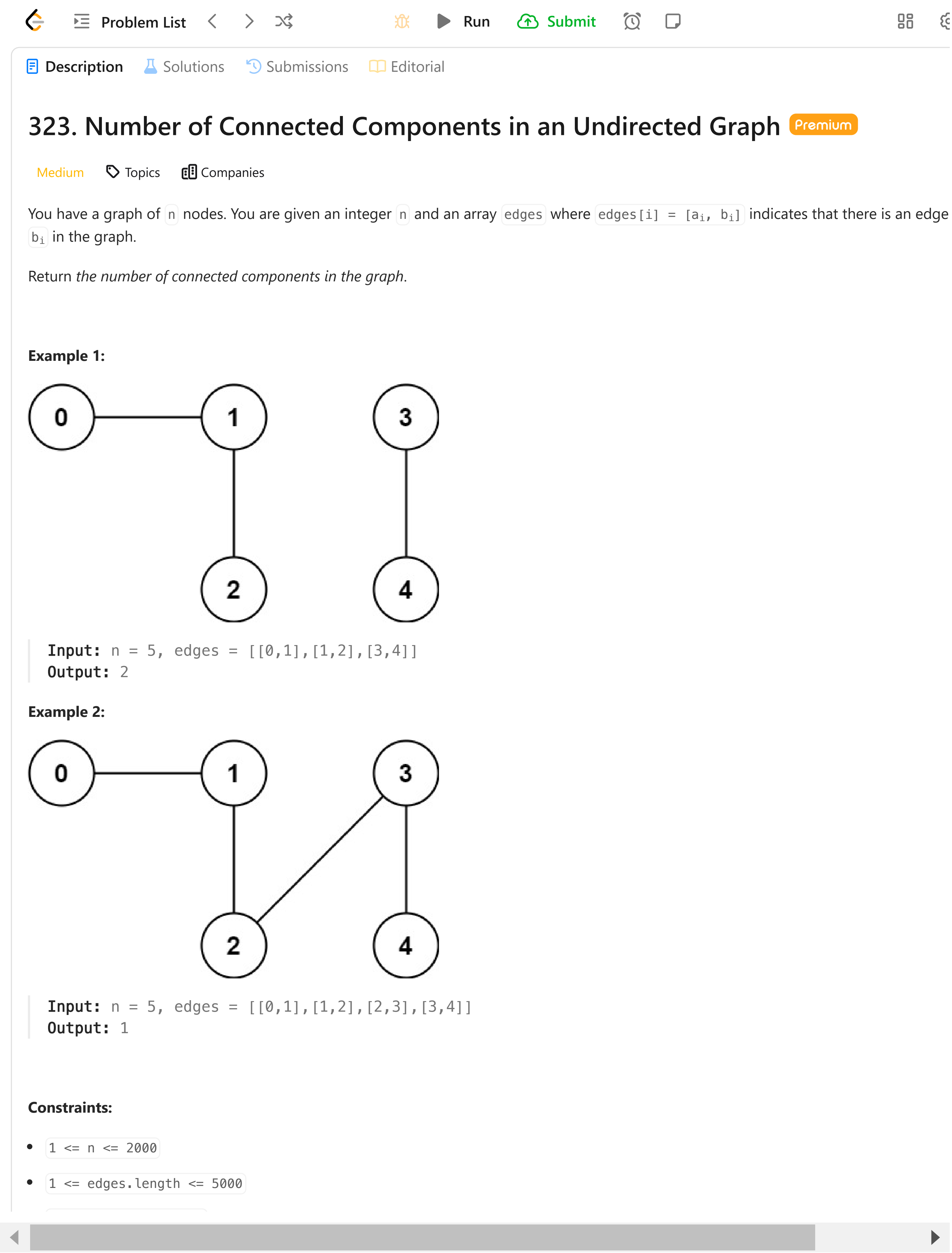
We also count how many building each '0' can be reached. It is stored in reach[][]. This can be done during the BFS. We also need to count how many total buildings are there in the matrix, which is stored in 'buildingNum'.

Finally, we can traverse the distance[][] matrix to get the point having shortest distance to all buildings. O(m*n)

The total time complexity will be O(m^2*n^2), which is quite high!. Please let me know if I did the analysis wrong or you have better solution.

```
public class Solution {
    public int shortestDistance(int[][] grid) {
        if (grid == null || grid[0].length == 0) return 0;
        final int[] shift = new int[] {0, 1, 0, -1, 0};

        int row  = grid.length, col = grid[0].length;
        int[][] distance = new int[row][col];
        int[][] reach = new int[row][col];
        int buildingNum = 0;

        for (int i = 0; i < row; i++) {
            for (int j =0; j < col; j++) {
                if (grid[i][j] == 1) {
                    buildingNum++;
                    Queue<int[]> myQueue = new LinkedList<int[]>();
                    myQueue.offer(new int[] {i,j});

                    boolean[][] isVisited = new boolean[row][col];
                    int level = 1;
```

```
public class Solution {
    public int shortestDistance(int[][] grid) {
        if (grid == null || grid[0].length == 0) return 0;
        final int[] shift = new int[] {0, 1, 0, -1, 0};

        int row  = grid.length, col = grid[0].length;
        int[][] distance = new int[row][col];
        int[][] reach = new int[row][col];
        int buildingNum = 0;

        for (int i = 0; i < row; i++) {
            for (int j =0; j < col; j++) {
                if (grid[i][j] == 1) {
                    buildingNum++;
                    Queue<int[]> myQueue = new LinkedList<int[]>();
                    myQueue.offer(new int[] {i,j});

                    boolean[][] isVisited = new boolean[row][col];
                    int level = 1;

                    while (!myQueue.isEmpty()) {
                        int qSize = myQueue.size();
                        for (int q = 0; q < qSize; q++) {
                            int[] curr = myQueue.poll();

                            for (int k = 0; k < 4; k++) {
                                int nextRow = curr[0] + shift[k];
                                int nextCol = curr[1] + shift[k + 1];

                                if (nextRow >= 0 && nextRow < row && nextCol >= 0 &&
nextCol < col
                                    && grid[nextRow][nextCol] == 0 &&
!isVisited[nextRow][nextCol]) {
                                        //The shortest distance from
[nextRow][nextCol] to thic building
                                        // is 'level'.
                                        distance[nextRow][nextCol] += level;
                                        reach[nextRow][nextCol]++;

                                        isVisited[nextRow][nextCol] = true;
                                        myQueue.offer(new int[] {nextRow, nextCol});
                                    }
                            }
                        }
                        level++;
                    }
                }
            }
        }

        int shortest = Integer.MAX_VALUE;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (grid[i][j] == 0 && reach[i][j] == buildingNum) {
                    shortest = Math.min(shortest, distance[i][j]);
                }
            }
        }

        return shortest == Integer.MAX_VALUE ? -1 : shortest;


    }
}
```



323. Number of Connected Components in an Undirected Graph

You have a graph of `n` nodes. You are given an integer `n` and an array `edges` where `edges[i] = [a_i, b_i]` indicates that there is an edge `b_i` in the graph.

Return *the number of connected components in the graph*.

Example 1:

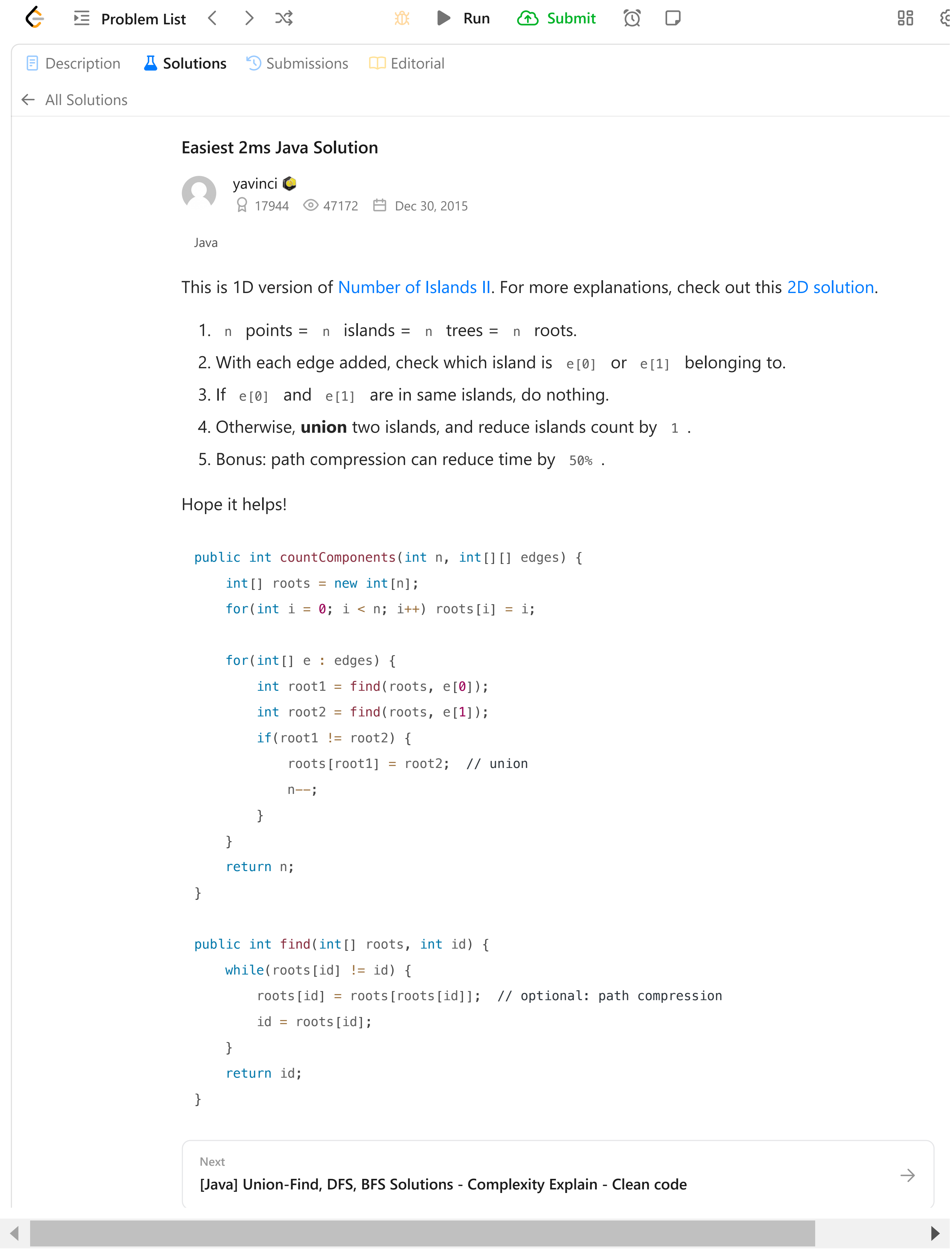
```
Input: n = 5, edges = [[0,1],[1,2],[3,4]]
Output: 2
```

Example 2:

```
Input: n = 5, edges = [[0,1],[1,2],[2,3],[3,4]]
Output: 1
```

Constraints:

- `1 <= n <= 2000`
- `1 <= edges.length <= 5000`



Easiest 2ms Java Solution

yavinci

17944 47172 Dec 30, 2015

Java

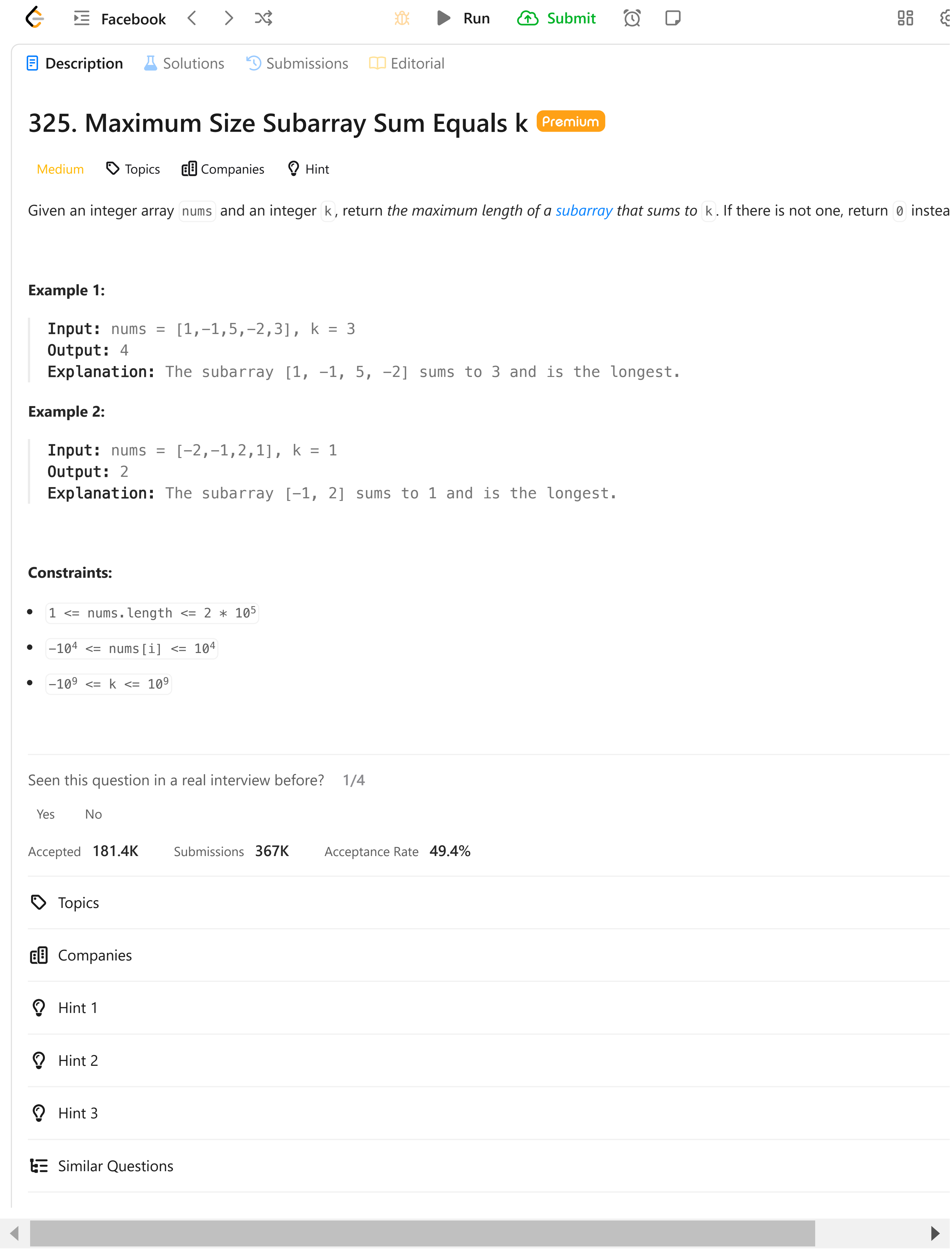
This is 1D version of Number of Islands II. For more explanations, check out this 2D solution.

1. `n` points = `n` islands = `n` trees = `n` roots.
2. With each edge added, check which island is `e[0]` or `e[1]` belonging to.
3. If `e[0]` and `e[1]` are in same islands, do nothing.
4. Otherwise, **union** two islands, and reduce islands count by `1` .
5. Bonus: path compression can reduce time by `50%` .

Hope it helps!

```java
public int countComponents(int n, int[][] edges) {
    int[] roots = new int[n];
    for(int i = 0; i < n; i++) roots[i] = i;

    for(int[] e : edges) {
        int root1 = find(roots, e[0]);
        int root2 = find(roots, e[1]);
        if(root1 != root2) {
            roots[root1] = root2;  // union
            n--;
        }
    }
    return n;
}

public int find(int[] roots, int id) {
    while(roots[id] != id) {
        roots[id] = roots[roots[id]];  // optional: path compression
        id = roots[id];
    }
    return id;
}
```

Next
[Java] Union-Find, DFS, BFS Solutions - Complexity Explain - Clean code



325. Maximum Size Subarray Sum Equals k

Given an integer array `nums` and an integer `k`, return *the maximum length of a* *subarray* *that sums to* `k`. If there is not one, return `0` instea

Example 1:

```
Input: nums = [1,-1,5,-2,3], k = 3
Output: 4
Explanation: The subarray [1, -1, 5, -2] sums to 3 and is the longest.
```

Example 2:

```
Input: nums = [-2,-1,2,1], k = 1
Output: 2
Explanation: The subarray [-1, 2] sums to 1 and is the longest.
```

Constraints:

- $1 <= nums.length <= 2 * 10^{5}$
- $-10^{4} <= nums[i] <= 10^{4}$
- $-10^{9} <= k <= 10^{9}$

Seen this question in a real interview before? 1/4

Yes No

Accepted 181.4K Submissions 367K Acceptance Rate 49.4%

Topics

Companies

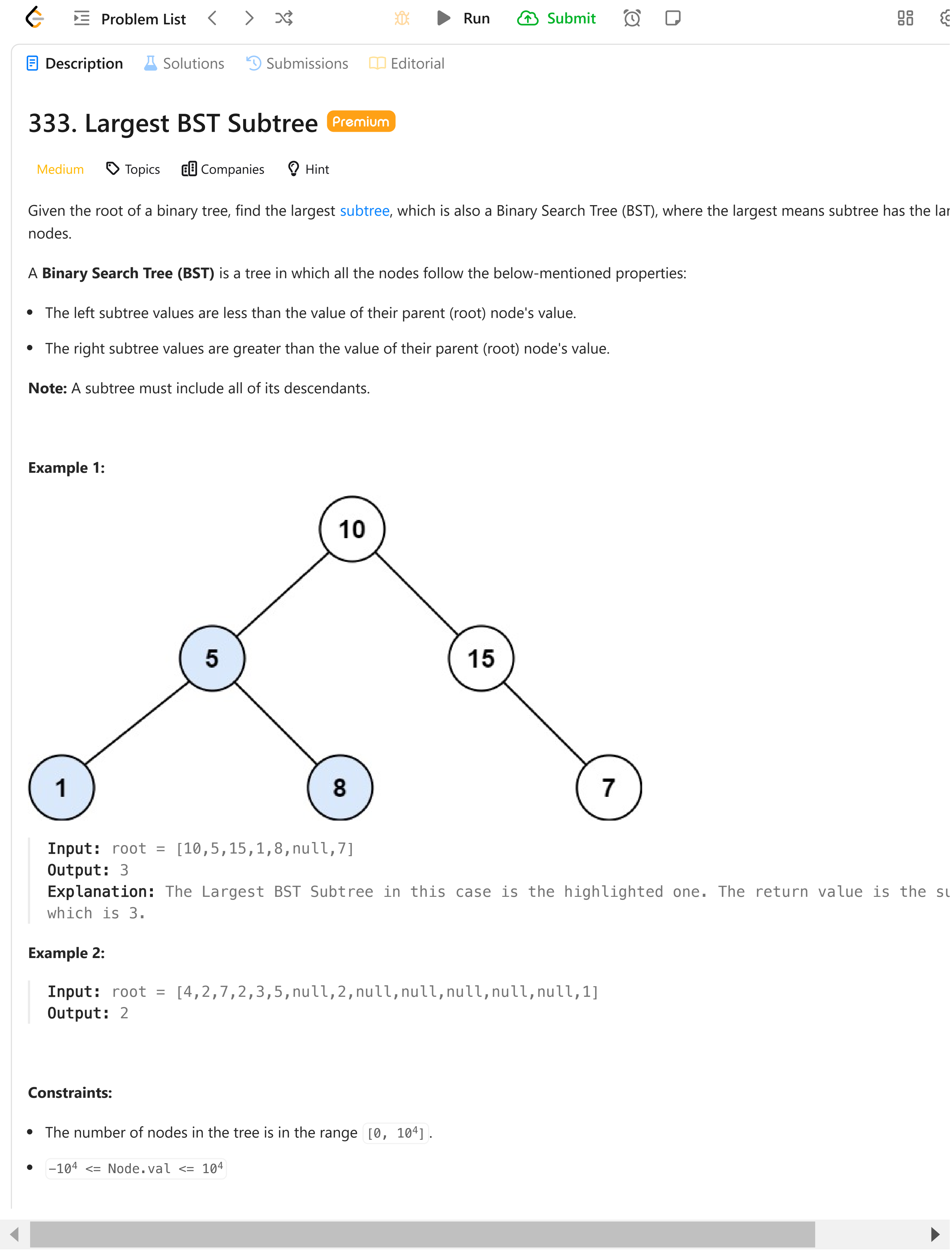
Hint 1

Hint 2

Hint 3

Similar Questions

```
    public int maxSubArrayLen(int[] nums, int k) {
        int currSum = 0, maxLen = 0; // set initial values for cumulative sum and
max length sum to k
        HashMap<Integer, Integer> sumToIndexMap = new HashMap<Integer, Integer>();
// key: cumulative sum until index i, value: i
        for (int i = 0; i < nums.length; i++) {
            currSum = currSum + nums[i]; // update cumulative sum

            // two cases where we can update maxLen
            if (currSum == k) maxLen = i + 1; // case 1: cumulative sum is k, update
maxLen for sure
            else if (sumToIndexMap.containsKey(currSum - k)) maxLen =
Math.max(maxLen, i - sumToIndexMap.get(currSum - k)); // case 2: cumulative sum is
more than k, but we can truncate a prefix of the array

            // store cumulative sum in map, only if it is not seen
            // because only the earlier (thus shorter) subarray is valuable, when we
want to get the maxLen after truncation
            if (!sumToIndexMap.containsKey(currSum)) sumToIndexMap.put(currSum, i);
        }
        return maxLen;
    }
```



333. Largest BST Subtree Premium

Medium Topics Companies Hint

Given the root of a binary tree, find the largest subtree, which is also a Binary Search Tree (BST), where the largest means subtree has the lar nodes.

A **Binary Search Tree (BST)** is a tree in which all the nodes follow the below-mentioned properties:

- The left subtree values are less than the value of their parent (root) node's value.
- The right subtree values are greater than the value of their parent (root) node's value.

Note: A subtree must include all of its descendants.

Example 1:

```
Input: root = [10,5,15,1,8,null,7]
Output: 3
Explanation: The Largest BST Subtree in this case is the highlighted one. The return value is the su
which is 3.
```

Example 2:

```
Input: root = [4,2,7,2,3,5,null,2,null,null,null,null,null,1]
Output: 2
```

Constraints:

- The number of nodes in the tree is in the range `[0, 10^4]`.
- `-10^4 <= Node.val <= 10^4`

```
public class Solution {

    class Result {  // (size, rangeLower, rangeUpper) -- size of current tree, range
of current tree [rangeLower, rangeUpper]
        int size;
        int lower;
        int upper;

        Result(int size, int lower, int upper) {
            this.size = size;
            this.lower = lower;
            this.upper = upper;
        }
    }

    int max = 0;

    public int largestBSTSubtree(TreeNode root) {
        if (root == null) { return 0; }
        traverse(root);
        return max;
    }

    private Result traverse(TreeNode root) {
        if (root == null) { return new Result(0, Integer.MAX_VALUE,
Integer.MIN_VALUE); }
        Result left = traverse(root.left);
        Result right = traverse(root.right);
        if (left.size == -1 || right.size == -1 || root.val <= left.upper ||
root.val >= right.lower) {
            return new Result(-1, 0, 0);
        }
        int size = left.size + 1 + right.size;
        max = Math.max(size, max);
        return new Result(size, Math.min(left.lower, root.val),
Math.max(right.upper, root.val));
    }
}
```