You can view this report online at : https://www.hackerrank.com/x/tests/1330879/candidates/42060038/report

Full Name:	Ruchit Bhardwaj
Email:	ruchitbh@usc.edu
Test Name:	**CodePath SE103: Unit 6 Assessment - Summer 2022**
Taken On:	12 Jul 2022 21:06:07 PDT
Time Taken:	76 min 46 sec/ 90 min
Personal Email Address:	ruchitbh@usc.edu
Invited by:	CodePath
Skills Score:	
Tags Score:	

100%

305/305

scored in **CodePath SE103: Unit 6 Assessment - Summer 2022** in 76 min 46 sec on 12 Jul 2022 21:06:07 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	DFS Graph Traversal › **Multiple Choice**	1 min 42 sec	5/ 5	✓
Q2	BFS Graph Traversal › **Multiple Choice**	4 min 20 sec	5/ 5	✓
Q3	DFS Bug › **Coding**	3 min 42 sec	75/ 75	✓
Q4	Walls and Gates › **Coding**	34 min 46 sec	80/ 80	✓
Q5	Connected Components in Undirected Graph › **Coding**	7 min 29 sec	70/ 70	✓
Q6	Graph Valid Tree › **Coding**	24 min 33 sec	70/ 70	✓

QUESTION 1

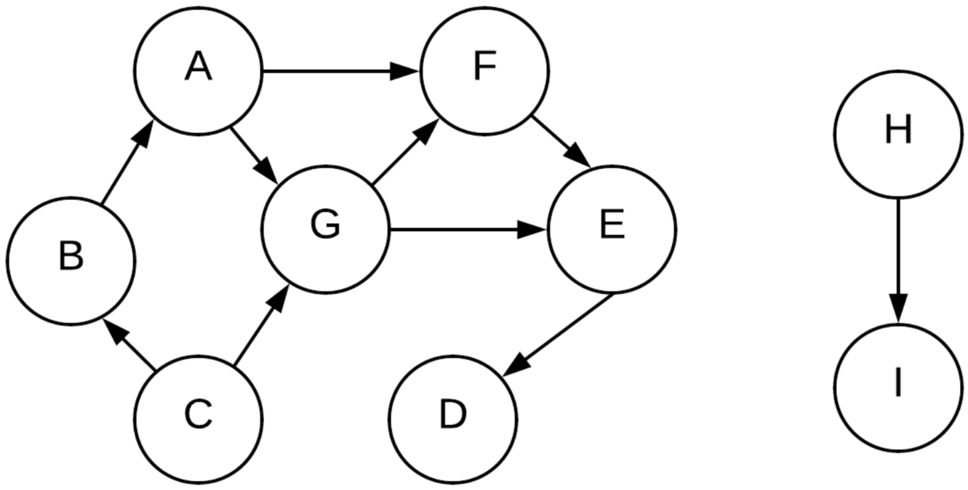
Correct Answer

Score 5

DFS Graph Traversal › Multiple Choice

QUESTION DESCRIPTION

Given this graph, answer the following questions:



What is the result of running preorder DFS starting on node C?
*Note: ties are broken alphabetically, so if node A had both node B and node C as neighbors, node B would be visited first.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

✓ [● CBAFEDG]
○ CGFEDBA
○ CBAGEDF
○ CGFEDBAHI

No Comments

QUESTION 2

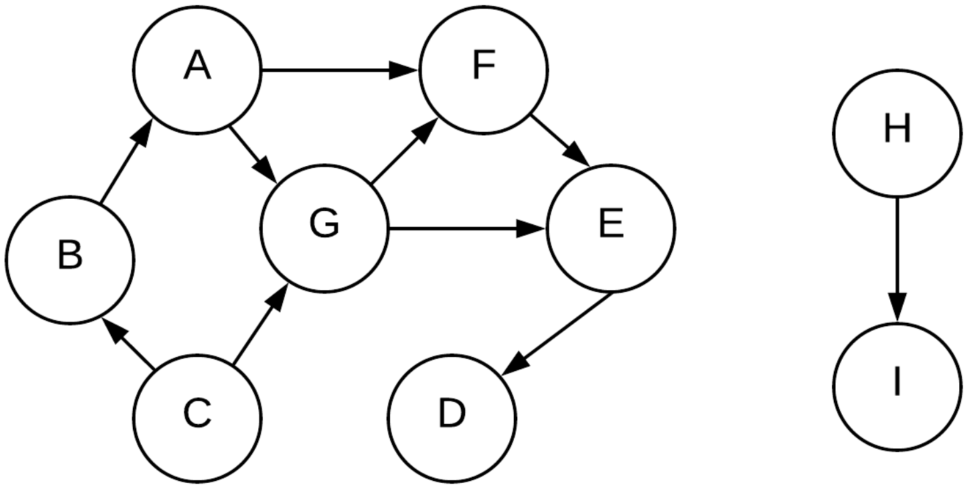
Correct Answer

Score 5

BFS Graph Traversal › Multiple Choice

QUESTION DESCRIPTION

Given this graph, answer the following questions:



What is the result of running BFS, using C as the root node?
*Note: ties are broken alphabetically, so if node A had both node B and node C as neighbors, node B would be added to the stack first

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- CBGAEFDHI
- [✓ CBGAEFD]
- CBGEFDAHI
- CBGEFDA

No Comments

QUESTION 3

Correct Answer

Score 75

DFS Bug › Coding

QUESTION DESCRIPTION

The following code is meant to run a DFS on a directed graph, but there's a bug. Fix this code snippet so that it is a proper DFS function!

If you're trying to understand how the test cases / inputs work, you can analyze the code outside of the function you're trying to implement to see how the input string is parsed to create the graph.

CANDIDATE ANSWER

Language used: **Java 8**

```
    /*
        Assuming this adjacency list graph structure and that a node with no
        Assuming this adjacency list graph structure and that a node with no
outgoing edges will not
        be included in the graph
        graph = {'A': ['B', 'C'],
                'B': ['D', 'E'],
                'C': ['F'],
                'E': ['F']}
    */
    public static ArrayList<String> dfs(HashMap<String, ArrayList<String>>
graph, String start) {
        ArrayList<String> visited = new ArrayList<String>();
        Stack<String> stack = new Stack<String>();
        stack.push(start);

        while(!stack.isEmpty()) {
            String vertex = stack.pop();
            if (!visited.contains(vertex)) {
                if (graph.containsKey(vertex)) {
                    ArrayList<String> neighbors = graph.get(vertex);
                    ArrayList<String> unvisited = new ArrayList<String>();
                    for (String n : neighbors) {
                        if (!visited.contains(n)) {
                            unvisited.add(n);
                        }
                    }
                    for (String s : unvisited)
                        stack.push(s);
                    visited.add(vertex);
                } else {
                    visited.add(vertex);
                }
            }
        }

        return visited;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	5	0.1549 sec	25.3 KB
Testcase 1	Easy	Hidden case	✓ Success	5	0.1052 sec	25 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.1546 sec	24.9 KB
Testcase 3	Easy	Hidden case	✓ Success	5	0.104 sec	25 KB
Testcase 4	Easy	Hidden case	✓ Success	5	0.0952 sec	24.9 KB
Testcase 5	Easy	Hidden case	✓ Success	5	0.0953 sec	25.1 KB
Testcase 6	Easy	Hidden case	✓ Success	5	0.0944 sec	24.9 KB
Testcase 7	Easy	Hidden case	✓ Success	5	0.1181 sec	25 KB
Testcase 8	Easy	Hidden case	✓ Success	5	0.0971 sec	25.2 KB
Testcase 9	Easy	Hidden case	✓ Success	5	0.1147 sec	24.9 KB
Testcase 10	Easy	Hidden case	✓ Success	5	0.1106 sec	25 KB
Testcase 11	Easy	Hidden case	✓ Success	5	0.1137 sec	24.9 KB
Testcase 12	Easy	Hidden case	✓ Success	5	0.1097 sec	25 KB
Testcase 13	Easy	Hidden case	✓ Success	5	0.1111 sec	24.9 KB
Testcase 14	Easy	Hidden case	✓ Success	5	0.1037 sec	25.1 KB

No Comments

QUESTION 4

Correct Answer

Score 80

Walls and Gates › Coding

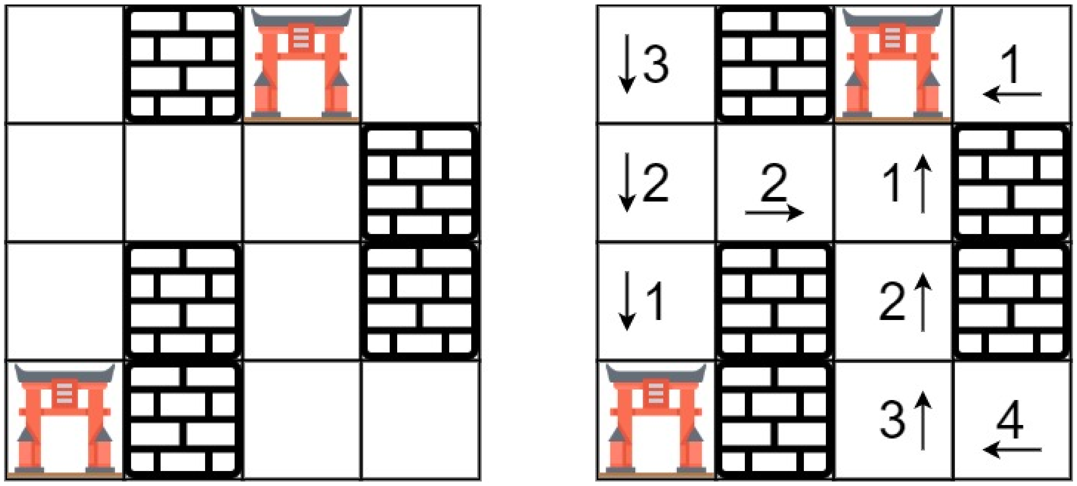
QUESTION DESCRIPTION

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 a gate is less than `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,-1,2147483647,-1],
[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]]
```

Example 3:

```
Input: rooms = [[2147483647]]
Output: [[2147483647]]
```

Example 4:

```
Input: rooms = [[0]]
Output: [[0]]
```

CANDIDATE ANSWER

Language used: **Java 8**

```
    public static void wallsAndGates(int[][] rooms ) {
        for (int i = 0; i < rooms.length; i++) {
            for (int j = 0; j < rooms[0].length; j++) {
                if (rooms[i][j] == 0) {
                    dfs(i, j, rooms, 0);
                }
            }
        }
    }

    public static void dfs(int i, int j, int[][] rooms, int count) {
        if (i < 0 || i >= rooms.length || j < 0 || j >= rooms[0].length ||
rooms[i][j] < count) {
            return;
        }
        rooms[i][j] = count;
        dfs(i + 1, j, rooms, count + 1);
        dfs(i - 1, j, rooms, count + 1);
        dfs(i, j + 1, rooms, count + 1);
        dfs(i, j - 1, rooms, count + 1);
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden case	✓ Success	10	0.1101 sec	24.8 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.1102 sec	24.8 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.1157 sec	24.9 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.1052 sec	25.2 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.1001 sec	25 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0979 sec	25.1 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.1029 sec	25.3 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.1277 sec	25.1 KB

No Comments

QUESTION 5

✓ Correct Answer

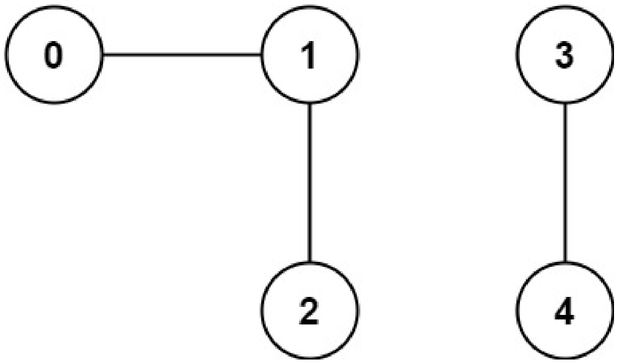
Score 70

Connected Components in Undirected Graph › Coding

QUESTION DESCRIPTION

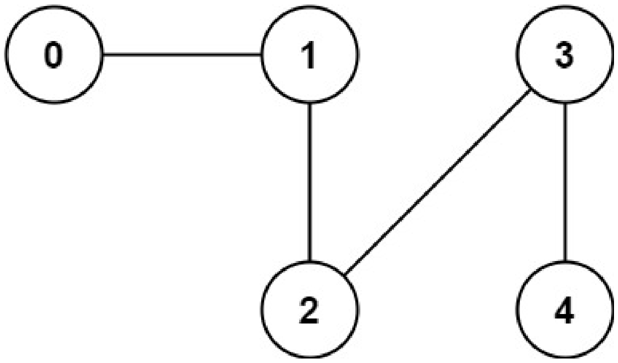
You have a graph of `n` nodes. You are given an integer `n` and an array `edges` where `edges[i] = [ai, bi]` indicates that there is an edge between `ai` and `bi` in the graph.
Return *the number of connected components in the graph*.

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
```

CANDIDATE ANSWER

Language used: **Java 8**

```
    public static int countComponents(int n, int[][] edges) {
        ArrayList<Edge>[] graph = new ArrayList[n];
        for (int i = 0; i < n; i++) {
            graph[i] = new ArrayList<>();
        }

        for (int[] edge: edges) {
            int u = edge[0];
            int v = edge[1];
            graph[u].add(new Edge(u, v));
            graph[v].add(new Edge(v, u));
        }

        boolean[] visited = new boolean[n];
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (!visited[i]) {
                getConnectedComponents(i, graph, visited, count);
                count++;
            }
        }
        return count;
    }

    public static void getConnectedComponents(int node, ArrayList<Edge>[]
graph, boolean[] visited, int count) {
        visited[node] = true;

        for (Edge e: graph[node]) {
            if (!visited[e.nbr]) {
                getConnectedComponents(e.nbr, graph, visited, count + 1);
            }
        }
    }

    public static class Edge {
        int node;
        int nbr;

        Edge(int node, int nbr) {
            this.node = node;
            this.nbr = nbr;
        }
    }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.106 sec	25.2 KB
Testcase 1	Easy	Hidden case	✓ Success	10	0.1031 sec	24.9 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1248 sec	25.1 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.1049 sec	24.8 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.091 sec	24.9 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.1133 sec	24.9 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.0975 sec	24.9 KB

No Comments

QUESTION 6

✓ Correct Answer

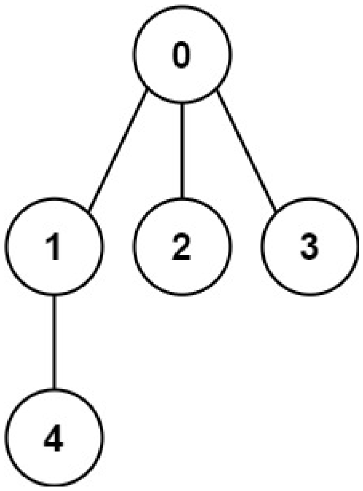
Score 70

Graph Valid Tree › Coding

QUESTION DESCRIPTION

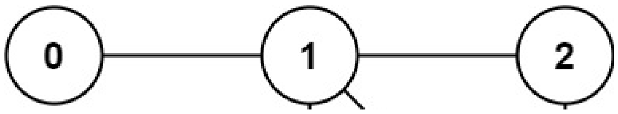
You have a graph of `n` nodes labeled from `0` to `n - 1`. You are given an integer n and a list of `edges` where `edges[i] = [`a_i`, `b_i`]` indicates that there is an undirected edge between nodes a_i and b_i in the graph.
Return `true` *if the edges of the given graph make up a valid tree, and* `false` *otherwise*.

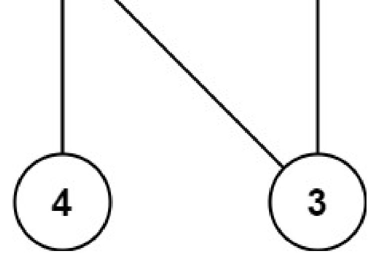
Example 1:



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

Example 2:





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

CANDIDATE ANSWER

Language used: **Java 8**

```java
    public static boolean validTree(int n, int[][] edges) {
        UnionFind uf = new UnionFind(n);
        for (int[] edge: edges) {
            if (!uf.union(edge[0], edge[1])) {
                return false;
            }
        }

        return uf.count() == 1;
    }

    public static class UnionFind {
        int[] parent;
        int[] rank;
        int count = 0;

        UnionFind(int n) {
            count = n;
            parent = new int[n];
            rank = new int[n];
            for (int i = 0; i < n; i++) {
                parent[i] = i;
            }
        }

        public int find(int n) {
            if (parent[n] == n) {
                return n;
            }
            return parent[n] = find(parent[n]);
        }

        public boolean union(int node1, int node2) {
            int u = find(node1);
            int v = find(node2);

            if (u == v) {
                return false;
            }

            if (rank[u] < rank[v]) {
                parent[u] = v;
                rank[v] += rank[u];
            } else if (rank[v] <= rank[u]) {
                parent[v] = u;
                rank[u] += rank[v];
            }
            count--;
            return true;
        }

        public int count() {
            return count;
        }
    }


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden case	✓ Success	10	0.1138 sec	25.1 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.1129 sec	25 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.0888 sec	25 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.1126 sec	24.8 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.1036 sec	24.9 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0903 sec	24.9 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.1088 sec	25.2 KB

No Comments

PDF generated at: 13 Jul 2022 05:25:17 UTC