You can view this report online at : https://www.hackerrank.com/x/tests/1121390/candidates/27361508/report

Full Name:	My Nguyen
Email:	nguyen_my@yahoo.com
Test Name:	**Graphs Assessment 2021**
Taken On:	5 Jul 2021 20:29:47 PDT
Time Taken:	76 min 12 sec/ 90 min
Personal Email Address:	nguyen_my@yahoo.com
Contact Number:	+14084096862
Resume:	https://hackerrank-resumes.s3.amazonaws.com/412894/JhbK9vK_4Bhc4Gvuv7s5hgcFJGeFCAThWIiNY1UGAfhwRPsrmVekT5ZtKXgX8QA2Ag/My_Nguyen_Resume.PDF
Linkedin:	https://www.linkedin.com/in/my-nguyen-87849
Invited by:	Curriculum
Skills Score:	
Tags Score:	

77%

235/305

scored in **Graphs Assessment 2021** in 76 min 12 sec on 5 Jul 2021 20:29:47 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	DFS Graph Traversal › **Multiple Choice**	1 min 56 sec	5/ 5	✓
Q2	BFS Graph Traversal › **Multiple Choice**	1 min 7 sec	5/ 5	✓
Q3	DFS Bug › **Coding**	5 min 2 sec	75/ 75	✓
Q4	Walls and Gates › **Coding**	33 min 45 sec	60/ 80	✓
Q5	Connected Components in Undirected Graph › **Coding**	22 min 58 sec	40/ 70	✓
Q6	Graph Valid Tree › **Coding**	10 min 51 sec	50/ 70	✓

QUESTION 1

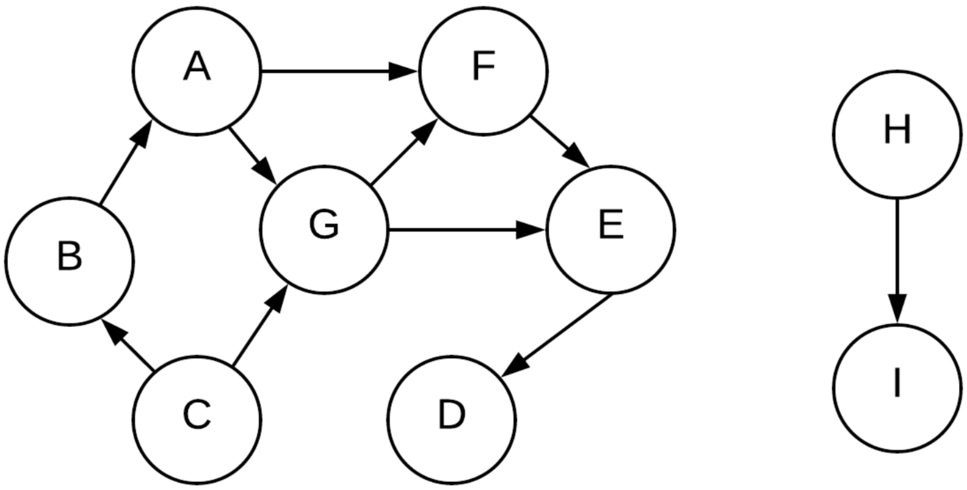
✓ Correct Answer

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

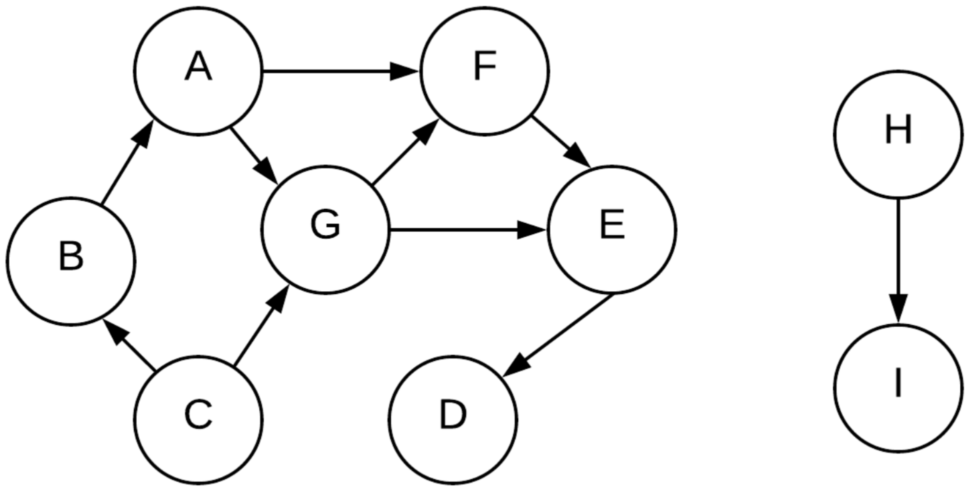
✓ Correct Answer

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
```

```
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	0.1204 sec	25 KB
Testcase 1	Easy	Hidden case	✓ Success	5	0.0833 sec	24.8 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.0883 sec	25 KB
Testcase 3	Easy	Hidden case	✓ Success	5	0.1393 sec	25.1 KB
Testcase 4	Easy	Hidden case	✓ Success	5	0.0785 sec	24.9 KB
Testcase 5	Easy	Hidden case	✓ Success	5	0.0991 sec	25 KB
Testcase 6	Easy	Hidden case	✓ Success	5	0.0939 sec	25 KB
Testcase 7	Easy	Hidden case	✓ Success	5	0.0848 sec	25 KB
Testcase 8	Easy	Hidden case	✓ Success	5	0.1403 sec	25 KB
Testcase 9	Easy	Hidden case	✓ Success	5	0.0913 sec	25 KB
Testcase 10	Easy	Hidden case	✓ Success	5	0.0796 sec	25 KB
Testcase 11	Easy	Hidden case	✓ Success	5	0.1154 sec	24.9 KB
Testcase 12	Easy	Hidden case	✓ Success	5	0.1053 sec	25.1 KB
Testcase 13	Easy	Hidden case	✓ Success	5	0.0861 sec	24.9 KB
Testcase 14	Easy	Hidden case	✓ Success	5	0.1099 sec	24.9 KB

No Comments

QUESTION 4

Correct Answer

Score 60

Walls and Gates › Coding

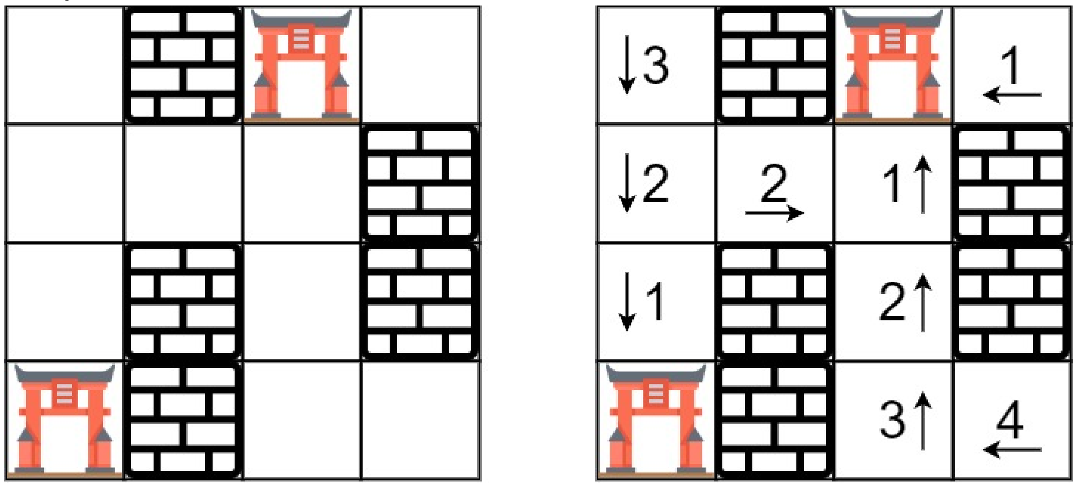
QUESTION DESCRIPTION

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 ) {
        List<int[]> gates = new ArrayList<>();
        for (int i = 0; i < rooms.length; i++) {
            for (int j = 0; j < rooms[i].length; j++) {
                if (rooms[i][j] == 0) {
                    gates.add(new int[]{i, j});
                }
            }
        }
        if (gates.isEmpty())
            return;

        Queue<int[]> queue = new LinkedList<>();
        for (int k = 0; k < gates.size(); k++) {
            queue.add(gates.get(k));
            Set<int[]> visited = new HashSet<>();
            int count = 0;
            while (!queue.isEmpty()) {
                int[] square = queue.poll();
                int i = square[0];
                int j = square[1];
                visited.add(new int[]{i, j});
                if (!gates.contains(square)) {
                    count++;

                    if (rooms[i][j] == 2147483647)
                        rooms[i][j] = count;
                    else
                        rooms[i][j] = Math.min(rooms[i][j], count);
                }

                // up
                int[] up = {i-1, j};
                if (i-1 >= 0 && rooms[i-1][j] != -1 && rooms[i-1][j] != 0 &&
!visited.contains(up)) {
                    queue.add(up);
                }
                // down
                int[] down = {i+1, j};
                if (i+1 < rooms.length && rooms[i+1][j] != -1 && rooms[i+1]
[j] != 0 && !visited.contains(down)) {
                    queue.add(down);
                }
                // left
                int[] left = {i, j-1};
                if (j-1 >= 0 && rooms[i][j-1] != -1 && rooms[i][j-1] != 0 &&
!visited.contains(left)) {
                    queue.add(left);
                }
                // right
                int[] right = {i, j+1};
                if (j+1 < rooms[i].length && rooms[i][j+1] != -1 && rooms[i]
[j+1] != 0 && !visited.contains(right)) {
                    queue.add(right);
                }
            }
        }
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden case	⊗ Terminated due to timeout	0	4.1399 sec	182 KB
Testcase 1	Easy	Sample	✓ Success	10	0.1403 sec	24.9 KB

		case				
Testcase 2	Easy	Sample case	✓ Success	10	0.0871 sec	24.9 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.0771 sec	25 KB
Testcase 4	Easy	Hidden case	⊗ Terminated due to timeout	0	4.0271 sec	231 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0794 sec	25.1 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.1433 sec	24.9 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.0797 sec	24.9 KB

No Comments

QUESTION 5

Correct Answer

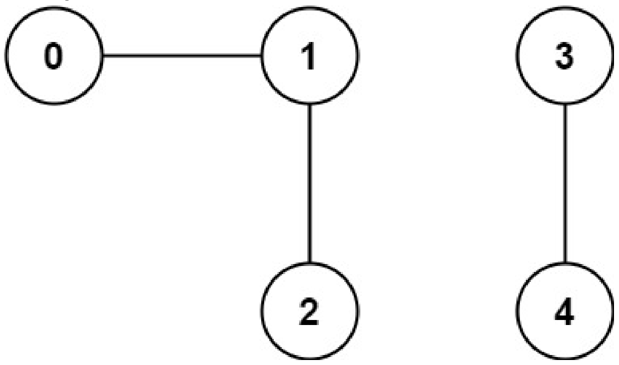
Score 40

Connected Components in Undirected Graph › Coding

QUESTION DESCRIPTION

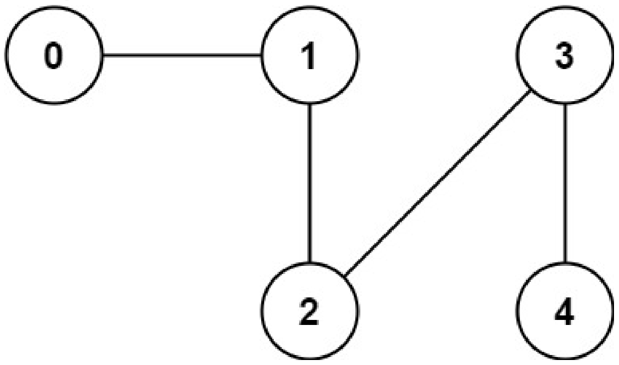
You have a graph of `n` nodes. You are given an integer `n` and an array `edges` where `edges[i] = [`a_i`, `b_i`]` indicates that there is an edge between a_i and b_i in the graph.
Return *the number of connected components in the graph*.

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
```

CANDIDATE ANSWER

Language used: **Java 8**

```java
    public static int countComponents(int n, int[][] edges) {
        Map<Integer, List<Integer>> map = new HashMap<>();
        for (int[] edge : edges) {
            int i = edge[0];
            int j = edge[1];
            List<Integer> listI = map.get(i);
            if (listI == null) {
                listI = new ArrayList<>();
                map.put(i, listI);
            }
            listI.add(j);
            List<Integer> listJ = map.get(j);
            if (listJ == null) {
                listJ = new ArrayList<>();
                map.put(j, listJ);
            }
            listJ.add(i);
        }

        boolean[] visited = new boolean[n];
        Stack<Integer> stack = new Stack<>();
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (!visited[i]) {
                stack.add(i);

                while (!stack.isEmpty()) {
                    int top = stack.pop();
                    if (visited[top])
                        continue;

                    visited[top] = true;
                    List<Integer> neighbors = map.get(top);
                    for (int neighbor : neighbors) {
                        if (!visited[neighbor]) {
                            stack.add(neighbor);
                        }
                    }
                }
                count++;
            }
        }
        return count;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0786 sec	24.9 KB
Testcase 1	Easy	Hidden case	✓ Success	10	0.1055 sec	25 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1519 sec	24.9 KB
Testcase 3	Easy	Hidden case	⊗ Runtime Error	0	0.1269 sec	25.1 KB
Testcase 4	Easy	Hidden case	⊗ Runtime Error	0	0.1324 sec	24.8 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.1056 sec	25 KB
Testcase 6	Easy	Hidden case	⊗ Runtime Error	0	0.1035 sec	24.9 KB

No Comments

QUESTION 6

✓ Correct Answer

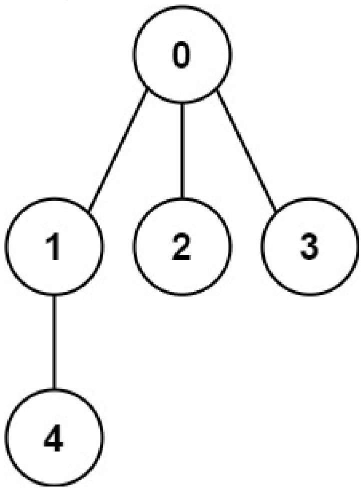
Score 50

Graph Valid Tree › Coding

QUESTION DESCRIPTION

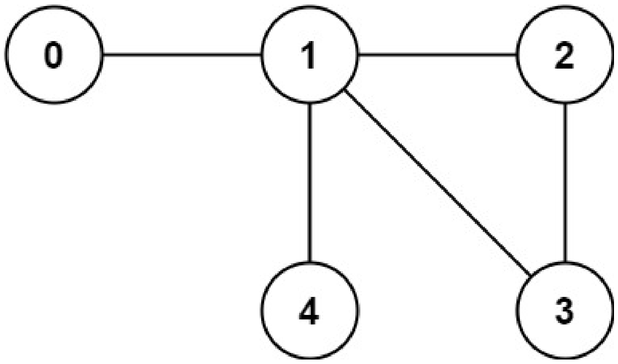
You have a graph of `n` nodes labeled from `0` to `n - 1`. You are given an integer n and a list of `edges` where `edges[i] = [`a_i`, `b_i`]` indicates that there is an undirected edge between nodes a_i and b_i in the graph.
Return `true` *if the edges of the given graph make up a valid tree, and* `false`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

```
    public static boolean validTree(int n, int[][] edges) {
        Map<Integer, List<Integer>> map = new HashMap<>();
        for (int[] edge : edges) {
            int i = edge[0];
            int j = edge[1];
            List<Integer> listI = map.get(i);
            if (listI == null) {
                listI = new ArrayList<>();
                map.put(i, listI);
            }
            listI.add(j);
            List<Integer> listJ = map.get(j);
            if (listJ == null) {
                listJ = new ArrayList<>();
                map.put(j, listJ);
            }
            listJ.add(i);
        }

        boolean[] visited = new boolean[n];
        Stack<Integer> stack = new Stack<>();
        for (int i = 0; i < n; i++) {
            if (!visited[i]) {
                stack.add(i);

                while (!stack.isEmpty()) {
                    int top = stack.pop();
                    if (visited[top])
                        continue;

                    visited[top] = true;
                    List<Integer> neighbors = map.get(top);
                    int count = 0;
                    for (int neighbor : neighbors) {
                        if (!visited[neighbor]) {
                            stack.add(neighbor);
                            count++;
                        }
                    }

                    if (count == 0) {
                        if (map.get(top).size() != 1)
                            return false;
                    }
                }
            }
        }
        return true;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden case	✓ Success	10	0.1507 sec	25 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0852 sec	24.9 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1036 sec	25 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.0834 sec	24.9 KB
Testcase 5	Easy	Hidden case	⊗ Runtime Error	0	0.0784 sec	24.8 KB
Testcase 5	Easy	Hidden case	⊗ Runtime Error	0	0.0961 sec	25.1 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.1091 sec	25 KB

No Comments