6154. Amount of Time for Binary Tree to Be Infected

My Submissions (/contest/weekly-contest-307/problems/amount-of-time-for-binary-tree-to-be-infected/submissions/)

Back to Contest (/contest/weekly-contest-307/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

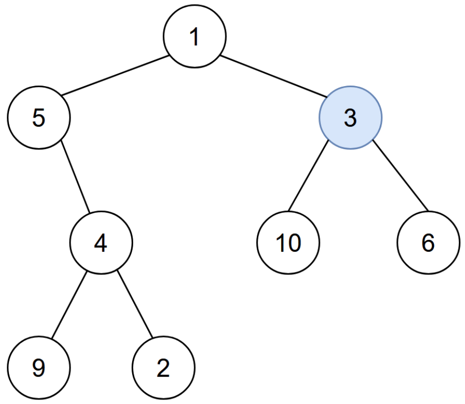
You are given the `root` of a binary tree with **unique** values, and an integer `start`. At minute `0`, an **infection** starts from the node with value `start`.

Each minute, a node becomes infected if:

- The node is currently uninfected.
- The node is adjacent to an infected node.

Return *the number of minutes needed for the entire tree to be infected.*

Example 1:



```
Input: root = [1,5,3,null,4,10,6,9,2], start = 3
Output: 4
Explanation: The following nodes are infected during:
- Minute 0: Node 3
- Minute 1: Nodes 1, 10 and 6
- Minute 2: Node 5
- Minute 3: Node 4
- Minute 4: Nodes 9 and 2
It takes 4 minutes for the whole tree to be infected so we return 4.
```

Example 2:

```
Input: root = [1], start = 1
Output: 0
Explanation: At minute 0, the only node in the tree is infected so we return 0.
```

Constraints:

- The number of nodes in the tree is in the range `[1, 10^5]`.
- `1 <= Node.val <= 10^5`
- Each node has a **unique** value.
- A node with a value of `start` exists in the tree.

JavaScript

```
const initializeGraphSet = (n) => { let g = []; for (let i = 0; i < n; i++) { g.push(new Set()); } return g; };

let g, n;
const amountOfTime = (root, start) => {
    n = 0;
    dfsGetMax(root);
    g = initializeGraphSet(n + 1);
    dfs(root);
    let dis = minDis(g, start), res = Number.MIN_SAFE_INTEGER;
    for (const x of dis) {
        if (x != Number.MAX_SAFE_INTEGER) res = Math.max(res, x);
    }
    return res;
};

const minDis = (g, start) => {
    let n = g.length, dis = Array(n).fill(Number.MAX_SAFE_INTEGER), q = [start];
    dis[start] = 0;
    while (q.length) {
        let cur = q.shift();
        for (const child of g[cur]) {
            if (dis[child] > dis[cur] + 1) {
                dis[child] = dis[cur] + 1;
                q.push(child);
            }
        }
    }
    return dis;
};

const dfs = (cur) => {
    if (!cur) return;
    if (cur.left) {
        g[cur.val].add(cur.left.val);
        g[cur.left.val].add(cur.val)
    }
    if (cur.right) {
        g[cur.val].add(cur.right.val);
        g[cur.right.val].add(cur.val)
    }
    n = Math.max(n, cur.val);
    dfs(cur.left);
    dfs(cur.right);
};

const dfsGetMax = (cur) => {
    if (!cur) return;
    n = Math.max(n, cur.val);
    dfsGetMax(cur.left);
    dfsGetMax(cur.right);
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/779154868/) More Details (/submissions/detail/779154868/)

Share your acceptance!

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)