6134. Find Closest Node to Given Two Nodes

My Submissions (/contest/weekly-contest-304/problems/find-closest-node-to-given-two-nodes/submissions/)

Back to Contest (/contest/weekly-contest-304/)

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

You are given a **directed** graph of `n` nodes numbered from `0` to `n - 1`, where each node has **at most one** outgoing edge.

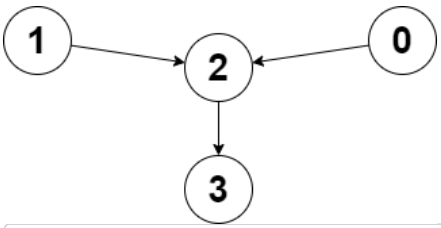
The graph is represented with a given **0-indexed** array `edges` of size `n`, indicating that there is a directed edge from node `i` to node `edges[i]`. If there is no outgoing edge from `i`, then `edges[i] == -1`.

You are also given two integers `node1` and `node2`.

Return *the **index** of the node that can be reached from both* `node1` *and* `node2`*, such that the **maximum** between the distance from* `node1` *to that node, and from* `node2` *to that node is **minimized***. If there are multiple answers, return the node with the **smallest** index, and if no possible answer exists, return `-1`.

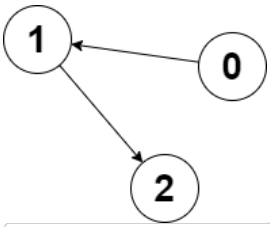
Note that `edges` may contain cycles.

Example 1:




```
Input: edges = [2,2,3,-1], node1 = 0, node2 = 1
Output: 2
Explanation: The distance from node 0 to node 2 is 1, and the distance from node 1 to node 2 is 1.
The maximum of those two distances is 1. It can be proven that we cannot get a node with a smaller maximum distance than 1, so
```

Example 2:




```
Input: edges = [1,2,-1], node1 = 0, node2 = 2
Output: 2
Explanation: The distance from node 0 to node 2 is 2, and the distance from node 2 to itself is 0.
The maximum of those two distances is 2. It can be proven that we cannot get a node with a smaller maximum distance than 2, so
```

Constraints:

- `n == edges.length`
- `2 <= n <= 10^5`
- `-1 <= edges[i] < n`
- `edges[i] != i`
- `0 <= node1, node2 < n`

JavaScript

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

```
2
3  const closestMeetingNode = (edges, node1, node2) => {
4      let n = edges.length, g = initializeGraph(n);
5      for (let i = 0; i < n; i++) {
6          if (edges[i] != -1) g[i].push(edges[i]);
7      }
8      let dis = bfs(g, node1), dis2 = bfs(g, node2), d = [];
9      for (let i = 0; i < n; i++) {
10         if (dis[i] != Number.MAX_SAFE_INTEGER & dis2[i] != Number.MAX_SAFE_INTEGER) {
11             d.push([Math.max(dis[i], dis2[i]), i]);
12         }
13     }
14     d.sort((x, y) => {
15         if (x[0] != y[0]) return x[0] - y[0];
16         return x[1] - y[1];
17     })
18     return d.length ? d[0][1] : -1;
19 };
20
21 const bfs = (g, start) => {
22     let n = g.length, dis = Array(n).fill(Number.MAX_SAFE_INTEGER), q = [start];
23     dis[start] = 0;
24     while (q.length) {
25         let cur = q.shift();
26         for (const child of g[cur]) {
27             if (dis[child] > dis[cur] + 1) {
28                 dis[child] = dis[cur] + 1;
29                 q.push(child);
30             }
31         }
32     }
33     return dis;
34 };
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/761130911/) More Details > (/submissions/detail/761130911/)

Share your acceptance!