6330. Shortest Cycle in a Graph

My Submissions (/contest/biweekly-contest-101/problems/shortest-cycle-in-a-graph/submissions/) Back to Contest (/contest/biweekly-contest-101/)

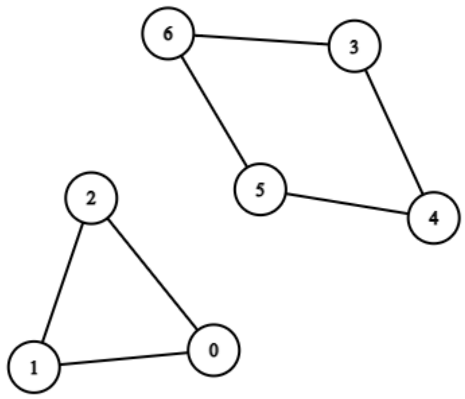
User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

There is a **bi-directional** graph with `n` vertices, where each vertex is labeled from `0` to `n - 1`. The edges in the graph are represented by a given 2D integer array `edges`, where `edges[i] = [`u_i`, `v_i`]` denotes an edge between vertex u_i and vertex v_i. Every vertex pair is connected by at most one edge, and no vertex has an edge to itself.

Return *the length of the **shortest** cycle in the graph*. If no cycle exists, return `-1`.

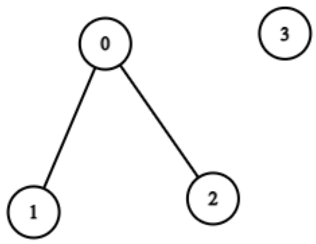
A cycle is a path that starts and ends at the same node, and each edge in the path is used only once.

Example 1:



```
Input: n = 7, edges = [[0,1],[1,2],[2,0],[3,4],[4,5],[5,6],[6,3]]
Output: 3
Explanation: The cycle with the smallest length is : 0 -> 1 -> 2 -> 0
```

Example 2:



```
Input: n = 4, edges = [[0,1],[0,2]]
Output: -1
Explanation: There are no cycles in this graph.
```

Constraints:

- `2 <= n <= 1000`
- `1 <= edges.length <= 1000`
- `edges[i].length == 2`
- $0 <= u_i, v_i < n$
- u_i != v_i
- There are no repeated edges.

JavaScript

```
const initializeGraph = (n) => { let g = []; for (let i = 0; i < n; i++) { g.push([]); } return g; };
const packUG = (g, edges) => { for (const [u, v] of edges) { g[u].push(v); g[v].push(u); } };

const findShortestCycle = (n, edges) => {
    let g = initializeGraph(n), res = Number.MAX_SAFE_INTEGER;
    packUG(g, edges);
    for (let i = 0; i < n; i++) res = Math.min(res, shortestCycleUG(g, i));
    return res == Number.MAX_SAFE_INTEGER ? -1 : res;
};

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

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/926082558/) More Details (/submissions/detail/926082558/)

Share your acceptance!

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

United States (/region)