5836. Number of Ways to Arrive at Destination

My Submissions (/contest/biweekly-contest-59/problems/number-of-ways-to-arrive-at-destination/submissions/)

Back to Contest (/contest/biweekly-contest-59/)

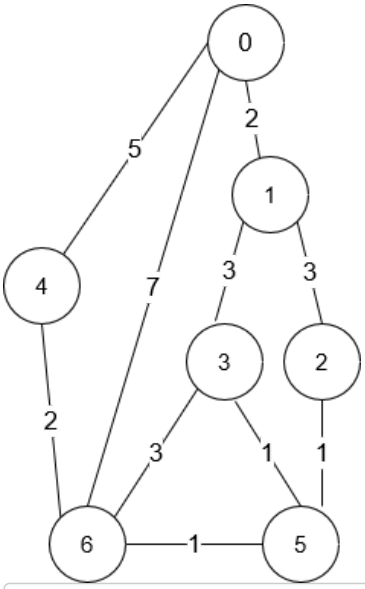
User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

You are in a city that consists of `n` intersections numbered from `0` to `n - 1` with **bi-directional** roads between some intersections. The inputs are generated such that you can reach any intersection from any other intersection and that there is at most one road between any two intersections.

You are given an integer `n` and a 2D integer array `roads` where `roads[i] = [u_i, v_i, time_i]` means that there is a road between intersections `u_i` and `v_i` that takes `time_i` minutes to travel. You want to know in how many ways you can travel from intersection `0` to intersection `n - 1` in the **shortest amount of time**.

Return *the **number of ways** you can arrive at your destination in the **shortest amount of time***. Since the answer may be large, return it **modulo** $10^9 + 7$.

Example 1:




```
Input: n = 7, roads = [[0,6,7],[0,1,2],[1,2,3],[1,3,3],[6,3,3],[3,5,1],[6,5,1],[2,5,1],[0,4,5],[4,6,2]]
Output: 4
Explanation: The shortest amount of time it takes to go from intersection 0 to intersection 6 is 7 minutes.
The four ways to get there in 7 minutes are:
- 0 → 6
- 0 → 4 → 6
- 0 → 1 → 2 → 5 → 6
- 0 → 1 → 3 → 5 → 6
```

Example 2:

```
Input: n = 2, roads = [[1,0,10]]
Output: 1
Explanation: There is only one way to go from intersection 0 to intersection 1, and it takes 10 minutes.
```

Constraints:

- `1 <= n <= 200`

- $n - 1 <= roads.length <= n * (n - 1) / 2$
- $roads[i].length == 3$
- $0 <= u_i, v_i <= n - 1$
- $1 <= time_i <= 10^9$
- $u_i != v_i$
- There is at most one road connecting any two intersections.
- You can reach any intersection from any other intersection.

JavaScript

```
const mod = 1e9 + 7;
const countPaths = (n, road) => {
    let adj = initializeGraph(n);
    for (const [u, v, cost] of road) {
        adj[u].push([v, cost]);
        adj[v].push([u, cost]);
    }
    return dijkstra(n, adj, 0);
};

const dijkstra = (n, g, source) => { // g: adjacent graph list, n: total vertices
    let dist = Array(n).fill(Number.MAX_SAFE_INTEGER);
    let ways = Array(n).fill(0);
    const pq = new MinPriorityQueue({ priority: x => x[0] * 200 + x[1] });
    dist[0] = 0;
    ways[0] = 1;
    pq.enqueue([0, source]);
    while (pq.size()) {
        let cur = pq.dequeue().element;
        let [curCost, curNode] = cur; // v: neighbour
        if (dist[curNode] != curCost) continue;
        for (const [node, cost] of g[curNode]) {
            let newDis = curCost + cost;
            if (newDis == dist[node]) {
                ways[node] += ways[curNode];
                ways[node] %= mod;
            } else if (newDis < dist[node]) {
                dist[node] = newDis;
                ways[node] = ways[curNode];
                pq.enqueue([dist[node], node]);
            }
        }
    }
    return ways[n - 1];
};

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

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/542087999/) | More Details > (/submissions/detail/542087999/)

Share your acceptance!