5941. Find All People With Secret

My Submissions (/contest/weekly-contest-269/problems/find-all-people-with-secret/submissions/) Back to Contest (/contest/weekly-contest-269/)

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

You are given an integer `n` indicating there are `n` people numbered from `0` to `n - 1`. You are also given a **0-indexed** 2D integer array `meetings` where `meetings[i] = [$x_i$, $y_i$, $time_i$]` indicates that person `$x_i$` and person `$y_i$` have a meeting at `$time_i$`. A person may attend **multiple meetings** at the same time. Finally, you are given an integer `firstPerson`.

Person `0` has a **secret** and initially shares the secret with a person `firstPerson` at time `0`. This secret is then shared every time a meeting takes place with a person that has the secret. More formally, for every meeting, if a person `$x_i$` has the secret at `$time_i$`, then they will share the secret with person `$y_i$`, and vice versa.

The secrets are shared **instantaneously**. That is, a person may receive the secret and share it with people in other meetings within the same time frame.

Return *a list of all the people that have the secret after all the meetings have taken place.* You may return the answer in **any order**.

Example 1:

```
Input: n = 6, meetings = [[1,2,5],[2,3,8],[1,5,10]], firstPerson = 1
Output: [0,1,2,3,5]
Explanation:
At time 0, person 0 shares the secret with person 1.
At time 5, person 1 shares the secret with person 2.
At time 8, person 2 shares the secret with person 3.
At time 10, person 1 shares the secret with person 5.
Thus, people 0, 1, 2, 3, and 5 know the secret after all the meetings.
```

Example 2:

```
Input: n = 4, meetings = [[3,1,3],[1,2,2],[0,3,3]], firstPerson = 3
Output: [0,1,3]
Explanation:
At time 0, person 0 shares the secret with person 3.
At time 2, neither person 1 nor person 2 know the secret.
At time 3, person 3 shares the secret with person 0 and person 1.
Thus, people 0, 1, and 3 know the secret after all the meetings.
```

Example 3:

```
Input: n = 5, meetings = [[3,4,2],[1,2,1],[2,3,1]], firstPerson = 1
Output: [0,1,2,3,4]
Explanation:
At time 0, person 0 shares the secret with person 1.
At time 1, person 1 shares the secret with person 2, and person 2 shares the secret with person 3.
Note that person 2 can share the secret at the same time as receiving it.
At time 2, person 3 shares the secret with person 4.
Thus, people 0, 1, 2, 3, and 4 know the secret after all the meetings.
```

Example 4:

```
Input: n = 6, meetings = [[0,2,1],[1,3,1],[4,5,1]], firstPerson = 1
Output: [0,1,2,3]
Explanation:
At time 0, person 0 shares the secret with person 1.
At time 1, person 0 shares the secret with person 2, and person 1 shares the secret with person 3.
Thus, people 0, 1, 2, and 3 know the secret after all the meetings.
```

Constraints:

- $2 <= n <= 10^5$
- $1 <= meetings.length <= 10^5$
- $meetings[i].length == 3$
- $0 <= x_i, y_i <= n - 1$
- $x_i != y_i$
- $1 <= time_i <= 10^5$
- $1 <= firstPerson <= n - 1$

JavaScript

```javascript
const initializeGraph = (n) => { let G = []; for (let i = 0; i < n; i++) { G.push([]); } return G; };
const packUGCost = (G, Edges) => { for (const [u, v, cost] of Edges) { G[u].push([v, cost]); G[v].push([u, cost]);
} };

const findAllPeople = (n, meetings, firstPerson) => {
    let g = initializeGraph(n), timeDis = Array(n).fill(Number.MAX_SAFE_INTEGER), res = [];
    packUGCost(g, meetings);
    let pq = new MinPriorityQueue({
        compare: (x, y) => {
            if (x[0] != y[0]) return x[0] - y[0];
            return x[1] - y[1];
        }
    });
    pq.enqueue([0, firstPerson]);
    pq.enqueue([0, 0]);
    timeDis[firstPerson] = 0;
    timeDis[0] = 0;
    while (pq.size()) {
        let [t, cur] = pq.dequeue();
        if (timeDis[cur] != t) continue;
        res.push(cur);
        for (const [child, tc] of g[cur]) {
            if (tc < t) continue;
            if (timeDis[child] > tc) {
                timeDis[child] = tc;
                pq.enqueue([tc, child]);
            }
        }
    }
    return res;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/593788091/) | More Details > (/submissions/detail/593788091/)

Share your acceptance!