2662. Minimum Cost of a Path With Special Roads

My Submissions (/contest/weekly-contest-343/problems/minimum-cost-of-a-path-with-special-roads/submissions/)

Back to Contest (/contest/weekly-contest-343/)

You are given an array start where start = [startX, startY] represents your initial position (startX, startY) in a 2D space. You are also given the array target where target = [targetX, targetY] represents your target position (targetX, targetY).

The cost of going from a position (x1, y1) to any other position in the space (x2, y2) is |x2 - x1| + |y2 - y2|y1|.

There are also some special roads. You are given a 2D array specialRoads where specialRoads [i] = $[x1_i, y1_i]$ $x2_i$, $y2_i$, $cost_i$] indicates that the ith special road can take you from $(x1_i, y1_i)$ to $(x2_i, y2_i)$ with a cost equal to cost; . You can use each special road any number of times.

Return the minimum cost required to go from (startX, startY) to (targetX, targetY).

User Accepted: 847 User Tried: 2903 Total Accepted: 905 **Total Submissions:** 7158 Difficulty: Medium

position/)

Example 1:

```
Input: start = [1,1], target = [4,5], specialRoads = [[1,2,3,3,2],[3,4,4,5,1]]
Output: 5
Explanation: The optimal path from (1,1) to (4,5) is the following:
-(1,1) \rightarrow (1,2). This move has a cost of |1-1| + |2-1| = 1.
- (1,2) -> (3,3). This move uses the first special edge, the cost is 2.
-(3,3) \rightarrow (3,4). This move has a cost of |3-3| + |4-3| = 1.
- (3,4) \rightarrow (4,5). This move uses the second special edge, the cost is 1.
So the total cost is 1 + 2 + 1 + 1 = 5.
It can be shown that we cannot achieve a smaller total cost than 5.
```

Example 2:

```
Input: start = [3,2], target = [5,7], specialRoads = [[3,2,3,4,4],[3,3,5,5,5],[3,4,5,6,6]]
Explanation: It is optimal to not use any special edges and go directly from the starting to the ending position with a cost
```

Constraints:

```
• start.length == target.length == 2
• 1 <= startX <= targetX <= 10<sup>5</sup>
• 1 <= startY <= targetY <= 10<sup>5</sup>
• 1 <= specialRoads.length <= 200
• specialRoads[i].length == 5

 startX <= x1<sub>i</sub>, x2<sub>i</sub> <= targetX</li>

 startY <= y1<sub>i</sub>, y2<sub>i</sub> <= targetY</li>
```

1 <= cost_i <= 10⁵

Discuss (https://leetcode.com/problems/minimum-cost-of-a-path-with-special-roads/discuss)

```
JavaScript
                                                                                                                      δ
                                                                                                                           C
1 \cdot \text{const minimumCost} = (a, b, c) \Rightarrow \{
        let m = new Map(), g = [];
        let A = build(m, g, a[0], a[1]), B = build(m, g, b[0], b[1]);
3
4
        g[A].push([B, cal(a[0], a[1], b[0], b[1])])
5 •
        for (const [x1, y1, x2, y2, cost] of c) {
6
            let u = build(m, g, x1, y1), v = build(m, g, x2, y2);
7
            g[A].push([u, cal(x1, y1, a[0], a[1])]);
            g[v].push([B, cal(x2, y2, b[0], b[1])]);
```

```
9
             g[u].push([v, cost]);
             for (const [xx, yy,] of c) {
10 •
11
                  let xxyy = build(m, g, xx, yy);
12
                  g[v].push([xxyy, cal(x2, y2, xx, yy)]);
13
14
         let d = dijkstra(g, A);
15
16
         return d[B];
17
    }
18
    const build = (m, g, x, y) \Rightarrow \{
let ke = x + "" + y;
19
20
21
         if (m.has(ke)) return m.get(ke);
22
         m.set(ke, g.length);
23
         g.push([]);
24
         return m.get(ke);
    };
25
26
27
    const cal = (x1, y1, x2, y2) \Rightarrow Math.abs(x1 - x2) + Math.abs(y1 - y2);
28
29 •
    const dijkstra = (g, start) \Rightarrow {
         let n = g.length, dis = Array(n).fill(Number.MAX_SAFE_INTEGER);
30
31 🕶
         let pq = new MinPriorityQueue({
32 ▼
             compare: (x, y) \Rightarrow \{
                  if (x[0] != y[0]) return x[0] - y[0];
33
34
                  return x[1] - y[1];
35
36
         });
37
         dis[start] = 0;
38
         pq.enqueue([0, start]);
39
         while (pq.size()) {
             let [d, cur] = pq.dequeue();
40
41
             if (d > dis[cur]) continue;
42 ▼
             for (const [child, cost] of g[cur]) {
43
                  let toChildCost = d + cost;
44 •
                  if (toChildCost < dis[child]) {</pre>
45
                      dis[child] = toChildCost;
46
                      pq.enqueue([toChildCost, child]);
47
                 }
48
             }
49
50
         return dis;
51
    };
```

□ Custom Testcase

Use Example Testcases

Submission Result: Accepted (/submissions/detail/945913485/) 2

More Details > (/submissions/detail/945913485/)

Run

♠ Submit

Share your acceptance!

Copyright © 2023 LeetCode

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

United States (/region)