2737. Find the Closest Marked Node Premium

Medium | Topics | Hint

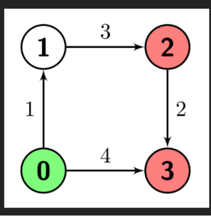
You are given a positive integer `n` which is the number of nodes of a **0-indexed directed weighted** graph and a **0-indexed 2D array** `edges` where `edges[i] = [u_i, v_i, w_i]` indicates that there is an edge from node u_i to node v_i with weight w_i.

You are also given a node `s` and a node array `marked`; your task is to find the **minimum** distance from `s` to **any** of the nodes in `marked`.

Return *an integer denoting the minimum distance from* `s` *to any node in* `marked` *or* `-1` *if there are no paths from s to any of the marked nodes.*

Example 1:

Input: n = 4, edges = [[0,1,1],[1,2,3],[2,3,2],[0,3,4]], s = 0, marked = [2,3]
Output: 4
Explanation: There is one path from node 0 (the green node) to node 2 (a red node), which is 0->1->2, and has a distance of 1 + 3 = 4.
There are two paths from node 0 to node 3 (a red node), which are 0->1->2->3 and 0->3, the first one has a distance of 1 + 3 + 2 = 6 and the second one has a distance of 4.
The minimum of them is 4.



1. defaultdict (list)

2. dijkstra

```python
class Solution:
    def minimumDistance(self, n: int, edges: List[List[int]], s: int, marked: List[int]) -> int:
        g = defaultdict(list)
        for u, v, w in edges:
            g[u].append((w, v)) #(weight, node)

        mh = [(0, s)]
        dis = [float("inf")] * n
        dis[s] = 0
        mark = set(marked)

        while mh:
            d, node = heapq.heappop(mh)
            if d > dis[node]:
                continue

            if node in mark:
                return d

            for nei_d, nei in g[node]:
                new_dis = d + nei_d
                if new_dis < dis[nei]:
                    dis[nei] = new_dis
                    heapq.heappush(mh, (new_dis, nei))

        return -1
```

→ 可以return immediately, 因为都是累计的

和Q1976 number of ways to arrive at destination 非常像.

都先转成graph, 然后dijkstra.