Problem 2699: Modify Graph Edge Weights

Problem Information

Difficulty: Hard
Acceptance Rate: 55.77%
Paid Only: No
Tags: Graph, Heap (Priority Queue), Shortest Path

Problem Description

You are given an **undirected weighted** **connected** graph containing `n` nodes labeled from `0` to `n - 1`, and an integer array `edges` where `edges[i] = [ai, bi, wi]` indicates that there is an edge between nodes `ai` and `bi` with weight `wi`.

Some edges have a weight of `-1` (`wi = -1`), while others have a **positive** weight (`wi > 0`).

Your task is to modify **all edges** with a weight of `-1` by assigning them **positive integer values** in the range `[1, 2 * 109]` so that the **shortest distance** between the nodes `source` and `destination` becomes equal to an integer `target`. If there are **multiple** **modifications** that make the shortest distance between `source` and `destination` equal to `target`, any of them will be considered correct.

Return _an array containing all edges (even unmodified ones) in any order if it is possible to make the shortest distance from_`source` _to_`destination` _equal to_`target` _, or an**empty array** if it's impossible._

Note: You are not allowed to modify the weights of edges with initial positive weights.

Example 1:

![](https://assets.leetcode.com/uploads/2023/04/18/graph.png)

Input: n = 5, edges = [[4,1,-1],[2,0,-1],[0,3,-1],[4,3,-1]], source = 0, destination = 1, target = 5 **Output:** [[4,1,1],[2,0,1],[0,3,3],[4,3,1]] **Explanation:** The graph above shows a possible modification to the edges, making the distance from 0 to 1 equal to 5.

Example 2:

![](https://assets.leetcode.com/uploads/2023/04/18/graph-2.png)

Input: n = 3, edges = [[0,1,-1],[0,2,5]], source = 0, destination = 2, target = 6 **Output:** [] **Explanation:** The graph above contains the initial edges. It is not possible to make the distance from 0 to 2 equal to 6 by modifying the edge with weight -1. So, an empty array is returned.

Example 3:

![](https://assets.leetcode.com/uploads/2023/04/19/graph-3.png)

Input: n = 4, edges = [[1,0,4],[1,2,3],[2,3,5],[0,3,-1]], source = 0, destination = 2, target = 6 **Output:** [[1,0,4],[1,2,3],[2,3,5],[0,3,1]] **Explanation:** The graph above shows a modified graph having the shortest distance from 0 to 2 as 6.

Constraints:

* `1 <= n <= 100` * `1 <= edges.length <= n * (n - 1) / 2` * `edges[i].length == 3` * `0 <= ai, bi < n` * `wi = -1 `or `1 <= wi <= 107` * `ai != bi` * `0 <= source, destination < n` * `source != destination` * `1 <= target <= 109` * The graph is connected, and there are no self-loops or repeated edges

Code Snippets

C++:

```
class Solution {
public:
vector<vector<int>> modifiedGraphEdges(int n, vector<vector<int>>& edges, int
source, int destination, int target) {

}
};
```

Java:

```
class Solution {
public int[][] modifiedGraphEdges(int n, int[][] edges, int source, int
destination, int target) {

}
}
```

Python3:

```
class Solution:
def modifiedGraphEdges(self, n: int, edges: List[List[int]], source: int,
destination: int, target: int) -> List[List[int]]:
```