Problem 1697: Checking Existence of Edge Length Limited Paths

Problem Information

Difficulty: Hard
Acceptance Rate: 63.05%
Paid Only: No
Tags: Array, Two Pointers, Union Find, Graph, Sorting

Problem Description

An undirected graph of `n` nodes is defined by `edgeList`, where `edgeList[i] = [ui, vi, disi]` denotes an edge between nodes `ui` and `vi` with distance `disi`. Note that there may be **multiple** edges between two nodes.

Given an array `queries`, where `queries[j] = [pj, qj, limitj]`, your task is to determine for each `queries[j]` whether there is a path between `pj` and `qj` such that each edge on the path has a distance **strictly less than** `limitj` .

Return _a**boolean array** _`answer` _, where_`answer.length == queries.length` _and the_`jth` _value of_`answer` _is_`true` _if there is a path for_`queries[j]`_is_`true` _, and_`false` _otherwise_.

Example 1:

![](https://assets.leetcode.com/uploads/2020/12/08/h.png)

Input: n = 3, edgeList = [[0,1,2],[1,2,4],[2,0,8],[1,0,16]], queries = [[0,1,2],[0,2,5]] **Output:** [false,true] **Explanation:** The above figure shows the given graph. Note that there are two overlapping edges between 0 and 1 with distances 2 and 16. For the first query, between 0 and 1 there is no path where each distance is less than 2, thus we return false for this query. For the second query, there is a path (0 -> 1 -> 2) of two edges with distances less than 5, thus we return true for this query.

Example 2:

![](https://assets.leetcode.com/uploads/2020/12/08/q.png)

Input: n = 5, edgeList = [[0,1,10],[1,2,5],[2,3,9],[3,4,13]], queries = [[0,4,14],[1,4,13]] **Output:** [true,false] **Explanation:** The above figure shows the given graph.

Constraints:

* `2 <= n <= 105` * `1 <= edgeList.length, queries.length <= 105` * `edgeList[i].length == 3` * `queries[j].length == 3` * `0 <= ui, vi, pj, qj <= n - 1` * `ui != vi` * `pj != qj` * `1 <= disi, limitj <= 109` * There may be **multiple** edges between two nodes.

Code Snippets

C++:

```
class Solution {
public:
vector<bool> distanceLimitedPathsExist(int n, vector<vector<int>>& edgeList,
vector<vector<int>>& queries) {

}
};
```

Java:

```
class Solution {
public boolean[] distanceLimitedPathsExist(int n, int[][] edgeList, int[][]
queries) {

}
}
```

Python3:

```
class Solution:
def distanceLimitedPathsExist(self, n: int, edgeList: List[List[int]],
queries: List[List[int]]) -> List[bool]:
```