5443. Find Critical and Pseudo-Critical Edges in Minimum Spanning Tree

:st/weekly-contest-194/problems/find-critical-and-pseudo-critical-edges-in-minimum-spanning-tree/submissions/)

st/weekly-contest-194/)

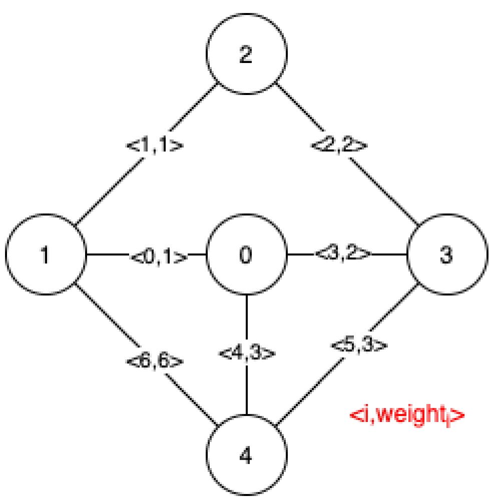
User Accepted:	0
User Tried:	2
Total Accepted:	0
Total Submissions:	2
Difficulty:	Hard

Given a weighted undirected connected graph with `n` vertices numbered from `0` to `n-1`, and an array `edges` where `edges[i] =` $[\texttt{from}_i, \texttt{to}_i, \texttt{weight}_i]$ represents a bidirectional and weighted edge between nodes $\texttt{from}_i$ and $\texttt{to}_i$. A minimum spanning tree (MST) is a subset of the edges of the graph that connects all vertices without cycles and with the minimum possible total edge weight.

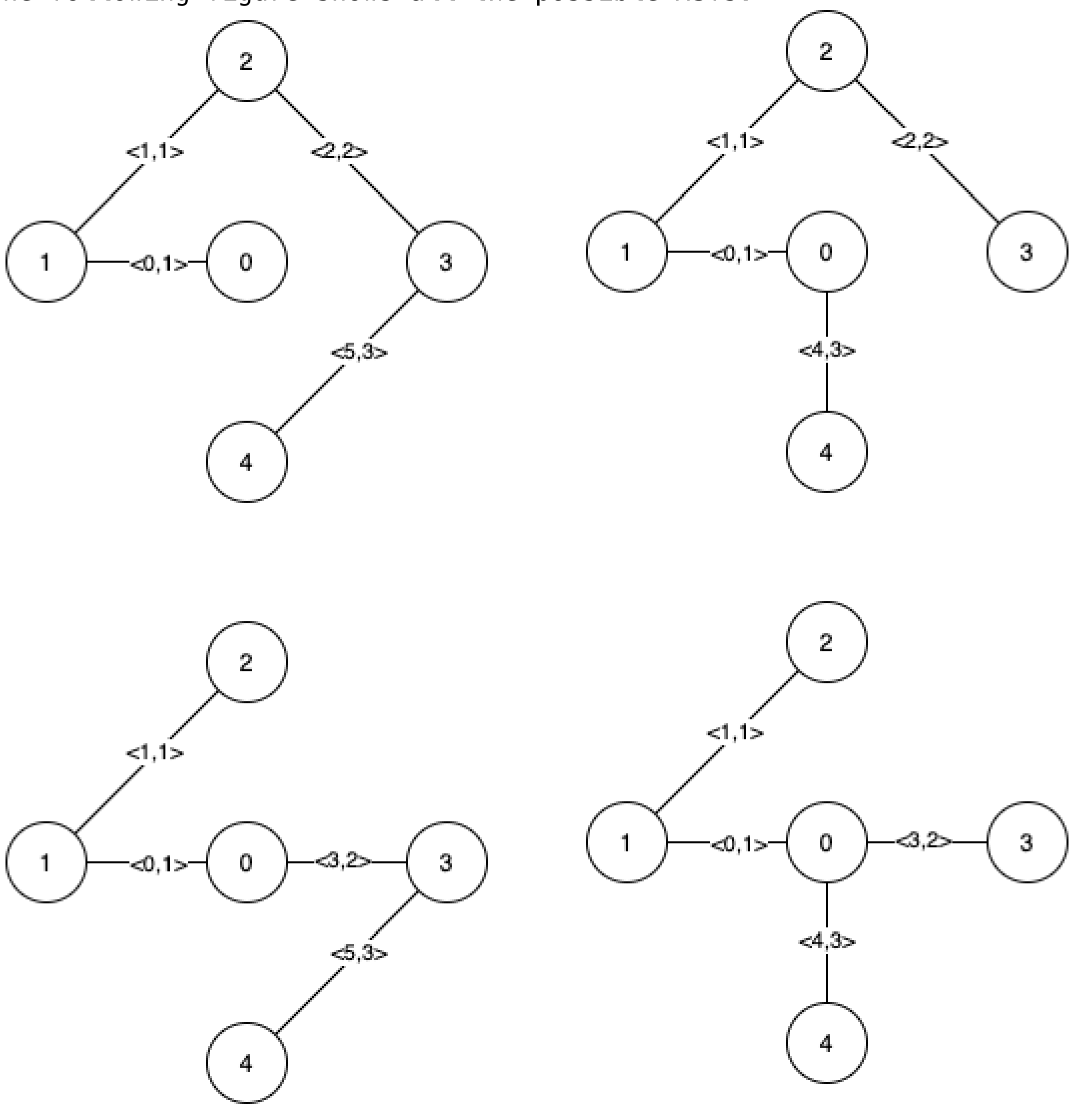
Find *all the critical and pseudo-critical edges in the minimum spanning tree (MST) of the given graph*. An MST edge whose deletion from the graph would cause the MST weight to increase is called a *critical edge*. A *pseudo-critical edge*, on the other hand, is that which can appear in some MSTs but not all.

Note that you can return the indices of the edges in any order.

Example 1:

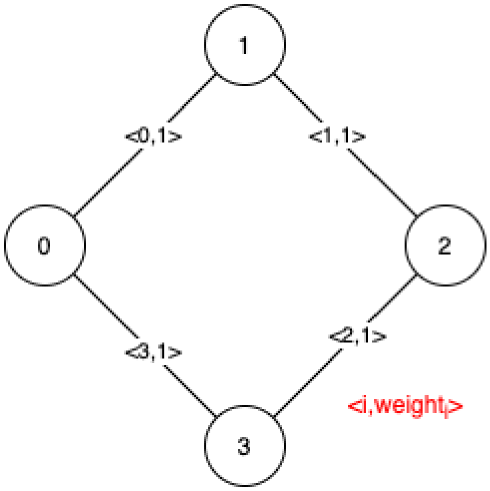


```
Input: n = 5, edges = [[0,1,1],[1,2,1],[2,3,2],[0,3,2],[0,4,3],[3,4,3],[1,4,6]]
Output: [[0,1],[2,3,4,5]]
Explanation: The figure above describes the graph.
The following figure shows all the possible MSTs:
```




```
Notice that the two edges 0 and 1 appear in all MSTs, therefore they are critical edges, s
The edges 2, 3, 4, and 5 are only part of some MSTs, therefore they are considered pseudo-
```

Example 2:




```
Input: n = 4, edges = [[0,1,1],[1,2,1],[2,3,1],[0,3,1]]
Output: [[],[0,1,2,3]]
Explanation: We can observe that since all 4 edges have equal weight, choosing any 3 edges
```

Constraints:

- `2 <= n <= 100`
- `1 <= edges.length <= min(200, n * (n - 1) / 2)`
- `edges[i].length == 3`
- `0 <= from`$_i$ `< to`$_i$ `< n`
- `1 <= weight`$_i$ `<= 1000`
- All pairs `(from`$_i$`, to`$_i$`)` are distinct.

JavaScript

```
/**
 * @param {number} n
 * @param {number[][]} edges
 * @return {number[][]}
 */
var findCriticalAndPseudoCriticalEdges = function(n, edges) {

};
```