Problem 1135: Connecting Cities With Minimum Cost

Problem Information

Difficulty: Medium
Acceptance Rate: 63.24%
Paid Only: Yes
Tags: Union Find, Graph, Heap (Priority Queue), Minimum Spanning Tree

Problem Description

There are `n` cities labeled from `1` to `n`. You are given the integer `n` and an array `connections` where `connections[i] = [xi, yi, costi]` indicates that the cost of connecting city `xi` and city `yi` (bidirectional connection) is `costi`.

Return _the minimum**cost** to connect all the _`n` _cities such that there is at least one path between each pair of cities_. If it is impossible to connect all the `n` cities, return `-1`,

The **cost** is the sum of the connections' costs used.

Example 1:

![](https://assets.leetcode.com/uploads/2019/04/20/1314_ex2.png)

Input: n = 3, connections = [[1,2,5],[1,3,6],[2,3,1]] **Output:** 6 **Explanation:** Choosing any 2 edges will connect all cities so we choose the minimum 2.

Example 2:

![](https://assets.leetcode.com/uploads/2019/04/20/1314_ex1.png)

Input: n = 4, connections = [[1,2,3],[3,4,4]] **Output:** -1 **Explanation:** There is no way to connect all cities even if all edges are used.

Constraints:

* `1 <= n <= 104` * `1 <= connections.length <= 104` * `connections[i].length == 3` * `1 <= xi, yi <= n` * `xi != yi` * `0 <= costi <= 105`

Code Snippets

C++:

```
class Solution {
public:
int minimumCost(int n, vector<vector<int>>& connections) {

}
};
```

Java:

```
class Solution {
public int minimumCost(int n, int[][] connections) {

}
}
```

Python3:

```
class Solution:
def minimumCost(self, n: int, connections: List[List[int]]) -> int:
```