**Connecting Cities With Minimum Cost**

**Difficulty: MEDIUM**

**Avg. time to solve**

**10 min**

**Success Rate**

**90%**

**Problem Statement**

**There are ‘N’ cities numbered from 1 to ‘N’ and ‘M’ roads. Each road connectss two different cities and described as a two-way road using three integers (‘U’, ‘V’, ‘W’) which represents the cost ‘W’ to connect city ‘U’ and city ‘V’ together.**

**Now, you are supposed to find the minimum cost so that for every pair of cities, there exists a path of connections (possibly of length 1) that connects those two cities together. The cost is the sum of the connection costs used. If the task is impossible, return -1.**

**Input Format :**

The first line of input contains an integer ‘T’ denoting the number of test cases. Then each test case follows.

The first line of each test case contains two single space-separated integers ‘N’ and ‘M’ denoting the number of cities and roads respectively.

Each of the next ‘M’ lines contains three single space-separated integers ‘U’, ‘V’, and ‘W’ denoting a two-way road between city ‘U’ and ‘V’ of cost ‘W’.

**Output Format :**

For each test case, return an integer denoting the minimum cost.

**Note :**

You don't need to print the output, it has already been taken care of. Just implement the given function.

**Constraints :**

1 <= T <= 50

1 <= N <= 10^4

1 <= M <= 10^4

1 <= W <= 10^3

1 <= U, V <= N

Time Limit: 1 sec

**Sample Input 1 :**

2

5 6

1 2 6

2 3 5

3 4 4

1 4 1

1 3 2

3 5 3

3 1

1 2 4

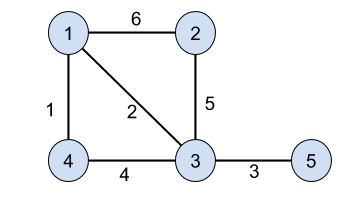
**Sample Output 1 :**

11

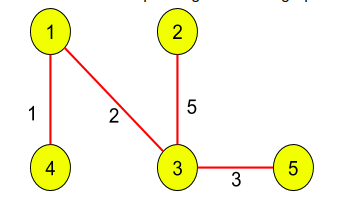
-1

**Explanation Of Sample Output 1 :**

For the first test case, the graph below describes the connection between the cities:



We can choose the following roads to connect all the cities getting minimum cost:



And its cost is 1 + 2 + 5 + 3 = 11

For the second test case, there is no way to connect all cities. So print -1.

**Sample Input 2 :**

2

3 3

1 2 1

2 3 2

3 1 3

4 3

1 2 4

2 3 5

3 4 1

**Sample Output 2 :**

3

10

**Explanation Of Sample Output 2 :**

For the first test case, the minimum cost will be 3.

For the second test case, the minimum cost will be 10.

#include<bits/stdc++.h>

void dfs(int i, vector<vector<pair<int, int>>> &graph, vector<bool> &visited) {

    visited[i]=true;

    for (auto x : graph[i]) {

        if (!visited[x.first]) dfs(x.first, graph, visited);

    }

}

int getMinCost(vector<vector<pair<int, int>>> &graph, int n) {

    priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

    vector<int> key(n, INT\_MAX), parent(n, -1);

    vector<bool> visited(n, false);

    int minCost=0;

    pq.push({0, 0});

    while (!pq.empty()) {

        pair<int, int> top=pq.top();

        pq.pop();

        if (parent[top.second]!=-1 and !visited[top.second]) minCost+=top.first;

        visited[top.second]=true;

        for (auto x : graph[top.second]) {

            if (!visited[x.first] and x.second<key[x.first]) {

                key[x.first]=x.second;

                parent[x.first]=top.second;

                pq.push({key[x.first], x.first});

            }

        }

    }

    return minCost;

}

int getMinimumCost(int n, int m, vector<vector<int>> &connections) {

  //  Write your code here.

    vector<vector<pair<int, int>>> graph(n);

    for (auto x : connections) {

        graph[x[0]-1].push\_back({x[1]-1, x[2]});

        graph[x[1]-1].push\_back({x[0]-1, x[2]});

    }

    vector<bool> visited(n, false);

    int count=0;

    for (int i=0; i<n; i++) {

        if (!visited[i]) {

            count++;

            dfs(i, graph, visited);

        }

    }

    if (count>1) return -1;

    return getMinCost(graph, n);

}