04. The Secret Network

Condition:

A spy organization maintains a secret communication network between its agents, spread across different cities. Each agent can send messages to another agent only if there is a direct connection between their cities. Each connection has a different cost to maintain, and some connections are so expensive that the organization's leadership wants to avoid them.

Your task is to help the organization create a network that connects all agents with minimal cost to maintain the connections. It is important to choose the optimal combination of connections so that each agent is connected to every other (directly or indirectly), but with minimal total cost.

Input:

1. The first line specifies two integers **N** and **M** ($1 \leq N \leq 1000$, $0 \leq M \leq 10000$), representing the number of cities and the number of connections between them.
2. Next are **M** lines, each of which contains three integers **A** , **B** , and **C** , which indicate that there is a connection between cities **A** and **B** with cost **C** .
3. The cities are numbered from 1 to **N.**

Output:

Output an integer that represents the minimum total cost of connecting all agents. If there is no way to connect all agents, output **-1** .

Additional conditions:

- Every connection is bidirectional, i.e. if there is a connection between cities **A** and **B** , it can be used in both directions.
- If the network is already connected, no new connections are required, only cost optimization.
- Some agents may be in isolated cities, meaning there are no connections to their city.

Example:

Input	Output
4 5 1 2 3 1 3 4 4 2 2 3 4 5	9

1 4 10	

Explanation:

The cheapest network to connect all agents is:

- Connection between cities 1 and 2 with cost 3.
- Connection between cities 4 and 2 with a cost of 2.
- Connection between cities 1 and 3 with a cost of 4.
  The total cost is 9.

Solving instructions:

- Represent the network as an undirected graph with weights on the edges.
- Use **Kruskal's algorithm** or **Prim's algorithm** to find the minimum spanning tree (MST) of the graph.
- Support a component union structure (Union-Find) for fast union and connectivity checking.
- If after running the algorithm not all cities are connected, output **-1** .