**Problem: Minimum Cost to Connect Cities**

**Scenario**

You are tasked with designing the most cost-efficient network to connect a group of cities. Each city is represented as a node, and each possible road between two cities has a certain construction cost. Your objective is to ensure that every city is connected to every other city either directly or indirectly, while minimizing the total construction cost.

**Problem Statement**

Given an integer n representing the number of cities and an array roads where each element is a tuple (city1, city2, cost) representing the cost to build a road between city1 and city2, find the minimum total cost to connect all cities. If it's not possible to connect all cities, return -1.

**Input Format**

* The first line contains an integer n, representing the number of cities.
* The second line contains an integer m, representing the number of possible roads.
* The next m lines each contain three integers city1, city2, and cost representing a road between city1 and city2 with a construction cost of cost.

**Constraints**

* 1 <= n <= 1000
* 0 <= m <= n \* (n - 1) / 2
* 1 <= cost <= 10^6

**Output Format**

* Print a single integer representing the minimum total cost to connect all cities, or -1 if it is not possible.

**Sample Input**

4

5

1 2 3

2 3 4

3 4 2

4 1 1

1 3 5

**Sample Output**

6

**Explanation**

The optimal way to connect all cities is to build the roads between:

* City 1 and City 4 with a cost of 1
* City 4 and City 3 with a cost of 2
* City 2 and City 1 with a cost of 3 Total cost = 1 + 2 + 3 = 6

**Solution**

The problem can be solved using Kruskal's Algorithm to find the Minimum Spanning Tree (MST) of the graph. If the number of edges in the MST is less than n-1, it means not all cities can be connected.

Here's the solution in Python:

python

class UnionFind:

def \_\_init\_\_(self, size):

self.parent = list(range(size))

self.rank = [0] \* size

def find(self, u):

if self.parent[u] != u:

self.parent[u] = self.find(self.parent[u])

return self.parent[u]

def union(self, u, v):

root\_u = self.find(u)

root\_v = self.find(v)

if root\_u != root\_v:

if self.rank[root\_u] > self.rank[root\_v]:

self.parent[root\_v] = root\_u

elif self.rank[root\_u] < self.rank[root\_v]:

self.parent[root\_u] = root\_v

else:

self.parent[root\_v] = root\_u

self.rank[root\_u] += 1

def minimumCostToConnectCities(n, roads):

uf = UnionFind(n + 1)

roads.sort(key=lambda x: x[2])

total\_cost = 0

edges\_used = 0

for u, v, cost in roads:

if uf.find(u) != uf.find(v):

uf.union(u, v)

total\_cost += cost

edges\_used += 1

if edges\_used == n - 1:

return total\_cost

return -1 if edges\_used < n - 1 else total\_cost

# Input reading

n = int(input())

m = int(input())

roads = []

for \_ in range(m):

u, v, cost = map(int, input().split())

roads.append((u, v, cost))

# Output

print(minimumCostToConnectCities(n, roads))

**Additional Test Cases**

**Test Case 1**

**Input:**

5

6

1 2 1

2 3 2

3 4 3

4 5 4

5 1 5

1 3 1

**Output:**

9

**Explanation:** The optimal way to connect all cities is to build the roads between:

* City 1 and City 2 with a cost of 1
* City 1 and City 3 with a cost of 1
* City 3 and City 4 with a cost of 3
* City 4 and City 5 with a cost of 4 Total cost = 1 + 1 + 3 + 4 = 9

**Test Case 2**

**Input:**

3

1

1 2 1

**Output:**

-1

**Explanation:** It is not possible to connect all cities with the given roads.

**Test Case 3**

**Input:**

6

10

1 2 1

2 3 2

3 4 3

4 5 4

5 6 5

6 1 6

1 3 1

3 5 1

2 4 2

4 6 2

**Output:**

7

**Explanation:** The optimal way to connect all cities is to build the roads between:

* City 1 and City 2 with a cost of 1
* City 1 and City 3 with a cost of 1
* City 3 and City 5 with a cost of 1
* City 2 and City 4 with a cost of 2
* City 4 and City 6 with a cost of 2 Total cost = 1 + 1 + 1 + 2 + 2 = 7

**Test Case 4**

**Input:**

4

4

1 2 3

2 3 4

3 4 5

4 1 6

**Output:**

12

**Explanation:** The optimal way to connect all cities is to build the roads between:

* City 1 and City 2 with a cost of 3
* City 2 and City 3 with a cost of 4
* City 3 and City 4 with a cost of 5 Total cost = 3 + 4 + 5 = 12

**Test Case 5**

**Input:**

2

1

1 2 1

**Output:**

1

**Explanation:** There is only one road, and it connects both cities. The total cost is 1.