Algorithm Design Assignment

Chosen Algorithm: Kruskal's Algorithm

I have chosen **Kruskal's Algorithm** because:

- **Graph Structure**: The given graph is **sparse** (fewer edges compared to the maximum possible edges). There are 10 cities (nodes) but only 12 edges.
- **Reason for Kruskal's Suitability**: Kruskal's algorithm performs better on sparse graphs since it focuses on sorting all the edges first and then greedily adding edges to the MST. This is more efficient than Prim's algorithm, which requires checking adjacent nodes repeatedly, making it less effective when the graph has fewer connections.

Justification for Rejecting Prim's Algorithm

- **Prim's Algorithm** performs better on dense graphs because it explores adjacent nodes through a priority queue, making it expensive to execute on sparse graphs.
- **Kruskal's Algorithm** works better when we know all the edges and want to build the MST from scratch, as in this problem, since the given graph is sparse, I chose to use this algorithm

Why Dynamic Programming or Divide and Conquer is not Suitable?

- These strategies are **inefficient** for MST problems:
    - **Dynamic Programming** solves subproblems repeatedly, which is unnecessary for MST as a greedy approach gives an optimal solution.
    - **Divide and Conquer** requires partitioning the problem into independent subproblems, but the MST problem needs a **global view of all edges** to ensure minimal total cost.

Pseudocode for Kruskal's Algorithm

```
Kruskal(Graph G):
  1. Sort all edges of G by weight in non-decreasing order.
  2. Initialize parent[] and rank[] arrays for union-find.
  3. Initialize an empty list MST to store the edges of the MST.
  4. for each edge (u, v, weight) in sorted edge list:
```

```
    if adding (u, v) does not form a cycle:
       Add (u, v) to MST.
       Perform union(u, v) to update the parent array.
5. Return MST and the total cost.
```

Complexity Analysis

- **Time Complexity**: (O(E \log E + E \log V)), where E is the number of edges and V is the number of nodes.
- **Space Complexity**: (O(V + E)) due to the storage for edges, parent, and rank arrays.

Comparison with Sequential Greedy Approach

The greedy approach (connecting cities in sequence) can result in higher costs and suboptimal connections. For example, connecting cities 0-1-2-3-... would not minimize costs.

Visualization of Partial MST (for each iteration)