1. Design and implement a Java program to determine the **Minimum Spanning Tree (MST)** of a connected, weighted, undirected graph using **Prim's Algorithm**. The program should allow the user to input the number of vertices and edges along with their corresponding weights, construct the graph using either an adjacency matrix or adjacency list representation, and then apply Prim's algorithm to select edges that form the MST with the **minimum total cost**. The program should display the edges chosen for the MST and their weights, as well as the total minimum cost of the spanning tree. After implementing the algorithm, analyze and find the **time complexity** of Prim's Algorithm.

2. Design and implement a Java program to find the **Minimum Spanning Tree (MST)** of a connected, weighted, undirected graph using **Kruskal's Algorithm**. The program should allow the user to input the number of vertices and edges along with their respective weights, store the edges in a suitable structure, and then apply Kruskal's algorithm to select the edges that form the MST with the **minimum total cost** while ensuring that no cycles are created. The program should display the edges selected for the MST along with their weights and the total cost of the spanning tree. After implementing the algorithm, analyze and find the **time complexity** of Kruskal's Algorithm.

3. Design and implement a Java program to find the **shortest path** from a given source vertex to all other vertices in a **weighted, connected graph** using **Dijkstra's Algorithm**. The program should allow the user to input the number of vertices, edges, and their weights, construct the graph using an adjacency matrix or adjacency list representation, and then apply Dijkstra's algorithm to compute the minimum distance from the source vertex to every other vertex. The program should display the shortest distance of each vertex from the source and the corresponding shortest paths. After implementing the algorithm, analyze and find the **time complexity** of Dijkstra's Algorithm.