Project Title: Graph Algorithms for Shortest Paths and Minimum Spanning Trees

Objective: The goal of this project is to implement and compare algorithms for finding shortest paths and minimum spanning trees in graphs.

Tasks:

1. **Implement Graph Data Structure:**
   - Create a data structure to represent a weighted graph. Include operations for adding vertices, edges, and weights.
2. **Choose Shortest Path Algorithms:**
   - Select at least two different algorithms for finding shortest paths (e.g., Dijkstra's Algorithm, Bellman-Ford Algorithm, Floyd-Warshall Algorithm, etc.).
3. **Choose Minimum Spanning Tree Algorithms:**
   - Select at least two different algorithms for finding minimum spanning trees (e.g., Prim's Algorithm, Kruskal's Algorithm, etc.).
4. **Generate Random Weighted Graphs:**
   - Write a function to generate random weighted graphs with varying sizes and structures.
5. **Measure Execution Time:**
   - Implement a timer to record the execution time of each shortest path and minimum spanning tree algorithm for different input graph sizes.
6. **Analyze Time Complexity:**
   - Based on the execution times, analyze and compare the time complexity of the algorithms. Consider factors like best case, worst case, and average case time complexity.
7. **Visualize the Data:**
   - Create plots or graphs to visually represent the execution times of the algorithms. This will help in demonstrating the differences in performance.
8. **Write a Report:**
   - Summarize your findings in a report. Include the following:
     - Introduction to graph algorithms for shortest paths and minimum spanning trees and their time complexity.
     - Explanation of the chosen algorithms.
     - Execution time results for different input graph sizes.
     - Analysis of time complexity.
     - Visual representations (graphs or plots).
     - Conclusions and observations.

Optional Enhancements:

1. **Compare Space Complexity:**
   - Extend the project to compare the space complexity of the algorithms.
2. **Apply to Real-world Graphs:**
   - Test the algorithms on real-world graphs (e.g., transportation networks, social networks) and analyze their performance.

3. **Optimization Techniques:**
    - Explore and implement optimization techniques for the selected algorithms to see how they affect performance.

Submission:

Submit the following:

1. Source code files implementing the graph data structure and algorithms for finding shortest paths and minimum spanning trees.
2. A report documenting the project, including findings and visualizations.

Note: Ensure to provide comprehensive comments in your code and use meaningful variable names for better readability and understanding.