Analyzing The Performance of The Functions In 1.1 and 1.2

Functions In 1.1(Dijkstra's Algorithm):

Time Complexity: The time Complexity of Dijkstra's Algorithm is O(V^2) but with a min-priority queue it drops down to O((V + E)log V). Here, The time complexity of Dijkstra's algorithm is O((V + E) log V), where V is the number of vertices and E is the number of edges. In the provided code, the modification with parameter k increases the time complexity, especially when k is close to V.

Space Complexity: The space complexity is O(V) for maintaining the priority queue and O(E) for the adjacency list.

Accuracy: Dijkstra's algorithm is accurate for non-negative edge weights, which is noticeable in the code.

Functions In 1.2(Bellman-Ford Algorithm):

Time Complexity: The time complexity of the Bellman-Ford algorithm is O(VE). It's less efficient compared to Dijkstra's algorithm, especially for dense graphs.

Space Complexity: Similar to Dijkstra's algorithm, the space complexity is O(V).

Accuracy: Bellman-Ford works correctly even with negative edge weights, but it's slower than Dijkstra's algorithm for non-negative weights.

Impact of Factors:

Graph Size: Both algorithms are affected by the size of the graph. As the number of vertices and edges increases, the execution time of both algorithms also increases.

Graph Density: Dense graphs have more edges, leading to longer execution times for both algorithms. Sparse graphs with fewer edges have shorter execution times.

Value of k: Higher values of k in the modified Dijkstra's algorithm increase the execution time, especially for large graphs. It's essential to choose an appropriate value of k based on the graph size and density to balance accuracy and performance.

Performance Comparison:

- Dijkstra's algorithm tends to perform better for graphs with non-negative edge weights, especially when the graph is sparse or when the value of k is relatively small.
- Bellman-Ford is suitable for graphs with negative edge weights but becomes inefficient for large graphs due to its higher time complexity.
- The code measures the execution time for both algorithms for different graph sizes, providing a practical comparison of their performance as we can see in the graph plot below.