Dijkstra's Algorithm

Overview

- **Purpose**: Find the shortest path from a source node to all other nodes in a graph with non-negative edge weights.
- **Applications**: Routing, network optimization, geographical mapping.

Approaches:

1. Basic Implementation with Adjacency Matrix

Time Complexity: $O(V^2)$
Space Complexity: $O(V^2)$

Algorithm Steps:

1. **Initialize Distances**: Set the distance to the source node as 0 and all other nodes as infinity.
2. **Find Minimum Distance Node**: Select the unvisited node with the smallest distance.
3. **Update Distances**: For the selected node, update the distances of its neighbors.
4. **Mark Node as Visited**: Mark the current node as visited and repeat until all nodes are visited.

2. Implementation with Adjacency List and Min-Heap (Priority Queue)

Time Complexity: $O((V + E) \log V)$
Space Complexity: $O(V + E)$

Algorithm Steps:

1. **Initialize Distances and Priority Queue**: Set initial distances and add the source node to the priority queue.
2. **Process Nodes**: Extract the node with the smallest distance from the queue, update distances for its neighbors, and reinsert updated nodes.
3. **Continue Until All Nodes are Processed**: Repeat until the priority queue is empty.