DIJKSTRA'S ALGORITHM

Explanation: Dijkstra's algorithm is used to find the shortest path between nodes in a graph with weighted edges. It utilizes a priority queue to always expand the least costly node first, ensuring the shortest path is found efficiently.

Key Points:

- Dijkstra's algorithm starts by initializing the distance to the start node as zero and all other nodes as infinity.
- A priority queue is used to keep track of the nodes to be explored, ordered by the shortest known distance.
- The algorithm iteratively selects the node with the smallest distance, updates the distances to its neighbors, and adds them to the priority queue if a shorter path is found.
- It maintains a set of visited nodes to avoid reprocessing.
- The process continues until the target node is reached, at which point the shortest path and its distance are known.

GRAPH REPRESENTATION

Explanation: The graph is represented using an unordered map of characters mapping to another unordered map of characters and integers, which allows efficient storage and retrieval of edge weights. This structure helps in quickly accessing the neighbors and their respective distances from any given node.

Key Points:

- The graph is defined as a vector of tuples, where each tuple contains two characters (nodes) and an integer (weight).

- An unordered map is used to store the graph, where each node maps to another unordered map of its neighbors and the corresponding edge weights.
- This representation allows for efficient lookups and updates during the execution of Dijkstra's algorithm.

PRIORITY QUEUE USAGE

Explanation: A priority queue is crucial in Dijkstra's algorithm to ensure that the node with the smallest known distance is processed first. This helps in efficiently finding the shortest path by always expanding the least costly node.

Key Points:

- The priority queue stores pairs of distances and nodes, ordered by the smallest distance.
- Nodes are added to the priority queue with their current shortest known distance.
- The node with the smallest distance is extracted from the priority queue for processing.
- The priority queue is updated whenever a shorter path to a node is found.

DISTANCE AND PATH TRACKING

Explanation: Dijkstra's algorithm keeps track of the shortest distances to each node and the path taken to reach each node. This information is used to reconstruct the shortest path once the target node is reached.

Key Points:

- An unordered map is used to store the shortest distance to each node from the start node.
- Another unordered map, called 'previous', is used to store the preceding node for each node in the shortest path.
- The distance to the start node is initialized to zero, and all other nodes are initialized to infinity.
- The path is reconstructed by backtracking from the target node using the 'previous' map.

EDGE RELAXATION

Explanation: Edge relaxation is the process of updating the shortest known distance to a node if a shorter path is found through another node. This is a key step in Dijkstra's algorithm to ensure that the shortest paths are correctly identified.

Key Points:

- For each node processed, the algorithm checks all its neighbors.
- If a shorter path to a neighbor is found, the distance to that neighbor is updated.
- The neighbor is then added to the priority queue with the updated distance.
- This process is repeated for all nodes until the shortest path to the target node is found.

TERMINATION CONDITION

Explanation: Dijkstra's algorithm terminates when the target node is reached, and the shortest path and its distance are known. The algorithm ensures that the shortest path is found by always expanding the least costly node first.

Key Points:

- The algorithm continues processing nodes from the priority queue until the target node is reached.
- Once the target node is reached, the shortest path and its distance are known.
- The path is reconstructed by backtracking from the target node using the 'previous' map.
- The final shortest path and its distance are returned as the result.