Optimizing Shortest Path Computation

Name: Ojas Jogdand  
Roll no: 23BDS039  
Branch: DSAI Inistitute: IIIT (Trible - I) Dharwad  
Mail: 23bds039@gmail.com

*Abstract 1* — Dijkstra's algorithm is a widely used method for finding the shortest path in a graph with non-negative edge weights.It has various applications such as in network routing, GPS navigation, and other applications requiring efficient shortest-path calculations.

# Introduction (DIJKSTRA’S ALGORITHM)

Dijkstra's algorithm is one of the most important and widely used algorithms for solving the shortest path problem in a weighted graph with non-negative edge weights. The algorithm finds the shortest path from a starting node to all other nodes in the graph. It employs a greedy approach by selecting the closest node to the source at each step, ensuring that the shortest path is found. This makes it ideal for applications in network routing, GPS systems, and other areas that require efficient pathfinding across weighted networks. The complexity of Dijkstra's algorithm depends on the implementation, with improvements like priority queues (min-heaps) making it more scalable for large graphs.

# Ease of Use (DIJKSTRA’S ALGORITHM)

Dijkstra’s algorithm is conceptually simple and easy to implement, especially when using a priority queue for node selection. However, for large graphs, careful attention is required to optimize its performance. Understanding the structure of the graph and choosing the appropriate data structures (such as adjacency matrices or lists) is essential for maximizing efficiency.

# Complexity *(*DIJKSTRA’S ALGORITHM*)*

**Time Complexity**: O(E log V)  
Where V is the number of vertices and E is the number of edges. This complexity arises from using a priority queue to extract the minimum distance vertex efficiently.

**Space Complexity**: O(V + E)  
The space complexity is dominated by the storage needed for the graph representation (adjacency list or matrix) and the distance array.

# Snippet and output *(*DIJKSTRA’S ALGORITHM*)*



#### 

# *Explation (*DIJKSTRA’S ALGORITHM*)*

***Text-file Handling***: The program reads and writes data in .txt format, using **BufferedReader** and **BufferedWriter** to handle file I/O efficiently.

***Graph Representation:***

Dijkstra's algorithm works on graphs represented by an adjacency list or matrix, where each edge is associated with a weight.

***Initialization***:

 Set the distance to the source node as 0 and all other nodes as infinity.

 Use a priority queue (min-heap) to always select the node with the minimum distance.

**Relaxation**:

For each adjacent node, if the current path through the node offers a shorter distance, update the shortest distance.

***Termination***:

The algorithm terminates when all nodes have been processed, and the shortest path from the source to all other nodes is known.

##### Acknowledgment *(Heading 5)*

Thanks to our Prof. Dr. Animesh Chaturvedi for giving us this opportunity and the required resources leading to the successful completion of the project .

##### References

1. Princeton University Algorithm

<https://algs4.cs.princeton.edu/home/>