IMPLEMENTING DIJKSTRA'S ALGORITHM IN JAVASCRIPT FOR ROUTE OPTIMIZATION

Executive Summary

This case study explores the implementation of Dijkstra's shortest path algorithm using JavaScript. The project walks students through the logic of graph representation, priority queue simulation, and distance updates. It also includes a visualization suggestion using HTML Canvas to deepen conceptual understanding. By tracing the algorithm step-by-step and explaining the logic in a browser-compatible language, students develop a clear grasp of how optimal pathfinding works in real-world applications like maps, games, and routing systems.

1. Introduction

Dijkstra's algorithm is widely used in computer science to determine the shortest path between nodes in a graph. Understanding this algorithm enhances students' foundation in data structures, particularly graphs and priority queues. JavaScript provides an accessible and interactive platform for demonstrating this algorithm in a web environment.

2. Problem Definition

The objective is to:

- Implement Dijkstra's algorithm in JavaScript
- Use an adjacency list to represent a weighted graph
- Calculate the shortest path from a source node to all other nodes
- Optionally visualize the process using browser-based rendering

3. Graph Representation

javascript

CopyEdit

const graph = {

```
  A: { B: 2, C: 4 },
  B: { A: 2, C: 1, D: 7 },
  C: { A: 4, B: 1, D: 3 },
  D: { B: 7, C: 3 }
};
```

Explanation:

- The graph is stored as a dictionary of dictionaries.
- Each key represents a node, and its value is an object showing connected nodes with weights.

4. Dijkstra's Algorithm in JavaScript

```
javascript
CopyEdit
function dijkstra(graph, start) {
  const distances = {};
  const visited = {};
  const previous = {};

  // Initialize all distances to infinity
  for (let node in graph) {
    distances[node] = Infinity;
    previous[node] = null;
  }
  distances[start] = 0;

  while (Object.keys(visited).length < Object.keys(graph).length) {
    let minNode = null;
```

```
    for (let node in distances) {
      if (!visited[node]) {
        if (minNode === null || distances[node] < distances[minNode]) {
          minNode = node;
        }
      }
    }

    for (let neighbor in graph[minNode]) {
      let newDist = distances[minNode] + graph[minNode][neighbor];
      if (newDist < distances[neighbor]) {
        distances[neighbor] = newDist;
        previous[neighbor] = minNode;
      }
    }

    visited[minNode] = true;
  }

  return { distances, previous };
}
```

5. Logic Explanation

- distances: Tracks shortest known distances from source
- visited: Ensures nodes are processed only once
- previous: Records the optimal path

- Main loop:
    - Selects unvisited node with the smallest distance
    - Updates distances for neighbors if shorter paths are found

6. Example Output

Using source = A:

javascript

CopyEdit

```
const result = dijkstra(graph, 'A');
console.log(result.distances);
// { A: 0, B: 2, C: 3, D: 6 }

console.log(result.previous);
// { A: null, B: 'A', C: 'B', D: 'C' }
```

Shortest path to D: A → B → C → D

7. Common Student Errors

- Treating graph as unweighted
- Forgetting to mark visited nodes
- Incorrectly updating the shortest distance
- Failing to backtrack using previous to reconstruct paths

8. Suggested Enhancements

- Add a **Min-Heap Priority Queue** to improve efficiency from $O(n^2)$ to $O(n \log n)$
- Use **HTML Canvas** to draw the graph and animate node visits

- Add path reconstruction function:

```
javascript
CopyEdit
function getPath(previous, end) {
  const path = [];
  while (end) {
    path.unshift(end);
    end = previous[end];
  }
  return path;
}
```

9. Conclusion

This case study demonstrates how JavaScript can be used not only for frontend development but also for teaching core algorithmic logic. Implementing Dijkstra's algorithm provides students with experience in graph handling, iterative logic, and optimization strategies essential for both academic and applied computing.

10. References

- E. W. Dijkstra (1959). *A note on two problems in connexion with graphs*
- MDN Web Docs – JavaScript Loops and Objects
- FreeCodeCamp – JavaScript Graph Algorithms
- GeeksforGeeks – Dijkstra's Algorithm Implementation