Floyd-Warshall Algorithm in DSA: Beginner to Advanced Guide

Subtitle: Learn how to find shortest paths between all pairs of nodes in a weighted graph efficiently.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Floyd-Warshall Algorithm is a **dynamic programming approach** to compute **shortest paths between all pairs of vertices** in a weighted graph. It works with **negative edges** but **no negative cycles**.

Step 1: Core Concepts

- Computes **all-pairs shortest paths**
- Uses a **matrix (V x V)** to store distances
- Iteratively updates distance through **intermediate vertices**
- **Time Complexity:** $O(V^3)$

Why it matters: Essential for dense graphs, multiple-source shortest paths, and route optimization.

Step 2: Graph Representation

- Use an **adjacency matrix**

```
let INF = 1e9;

let dist = [

  [0, 3, INF, 7],

  [8, 0, 2, INF],

  [5, INF, 0, 1],

  [2, INF, INF, 0]

];
```

- INF represents no direct edge between nodes

Step 3: Implementation in JavaScript

```
function floydWarshall(dist){

  let V = dist.length;
```

```
  for(let k=0; k<V; k++){
    for(let i=0; i<V; i++){
      for(let j=0; j<V; j++){
        if(dist[i][k] + dist[k][j] < dist[i][j]){
          dist[i][j] = dist[i][k] + dist[k][j];
        }
      }
    }
  }
  return dist;
}

// Example usage
let result = floydWarshall(dist);
console.log(result);
/*
[
 [0, 3, 5, 6],
 [5, 0, 2, 3],
 [3, 6, 0, 1],
 [2, 5, 7, 0]
]
*/
```

Step 4: Applications

1. **All-pairs shortest path problems**
2. **Network routing and optimization**
3. **Transitive closure of graphs**

4. **Competitive programming: dense graphs**

Step 5: Mini Exercises

1. Implement Floyd-Warshall on a weighted adjacency matrix
2. Detect negative cycles using diagonal check (dist[i][i] < 0)
3. Solve grid-based shortest path problems using Floyd-Warshall
4. Compare results with multiple Dijkstra runs

Step 6: Common Mistakes

- Forgetting to handle INF + x correctly
- Using adjacency list → slower implementation
- Forgetting diagonal initialization (dist[i][i] = 0)
- Not checking for negative cycles in final matrix

Step 7: Practice Plan for Floyd-Warshall

Day 1:

- Implement basic Floyd-Warshall → 3–5 problems

Day 2:

- Detect negative cycles → 3–5 problems

Day 3:

- All-pairs shortest path applications → 5–7 problems

Key Takeaways

- Floyd-Warshall computes **shortest paths between all pairs**
- Handles **negative edges** but not negative cycles
- Works well for **dense graphs and multi-source shortest path problems**
- Practice improves **DP-on-graphs and matrix update logic**

Visit haas.dev for more Floyd-Warshall guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app