Floyd-Warshall Algorithm: A Dynamic Programming Approach

June 18, 2025

Abstract

The Floyd-Warshall algorithm is a dynamic programming technique for computing the shortest paths between all pairs of vertices in a weighted graph. Applicable to both directed and undirected graphs, it handles negative edge weights, provided no negative cycles exist. This document details the algorithm's formulation, pseudocode, complexity analysis, practical applications, and extensions, including negative cycle detection and path reconstruction. Additional insights into optimization and implementation considerations are provided for a comprehensive understanding.

1 Introduction

The Floyd-Warshall algorithm computes the **shortest paths between all pairs** of vertices in a weighted graph, represented by an adjacency matrix. It is a cornerstone of graph theory, leveraging **dynamic programming** to iteratively improve path distances by considering intermediate vertices. Unlike Dijkstra's or Bellman-Ford algorithms, it solves the all-pairs shortest path problem in a single run, making it ideal for dense graphs and applications requiring frequent distance queries.

2 Problem Formulation

Given a weighted graph $G = (V, E)$ with n vertices and edge weights $w(u, v)$, compute the shortest path distances between all pairs of vertices (u, v). The graph may have negative edge weights but must not contain negative weight cycles.

The output is a distance matrix $\text{dist}[i][j]$, where $\text{dist}[i][j]$ represents the shortest path distance from vertex i to vertex j.

2.1 Recurrence Relation

The algorithm uses the following dynamic programming recurrence:

$$\text{dist}_k[i][j] = \min(\text{dist}_{k-1}[i][j], \text{dist}_{k-1}[i][k] + \text{dist}_{k-1}[k][j])$$

where $\text{dist}_k[i][j]$ is the shortest path from i to j using only vertices $\{1, 2, \ldots, k\}$ as intermediates.

Initially:

$$\text{dist}_0[i][j] = \begin{cases} 0 & \text{if } i = j \\ w(i,j) & \text{if edge } (i,j) \text{ exists} \\ \infty & \text{otherwise} \end{cases}$$

3 Dynamic Programming Algorithm

The Floyd-Warshall algorithm constructs the distance matrix iteratively. Below is the pseudocode:

Algorithm 1 Floyd-Warshall Algorithm

```
Input: Graph G with n vertices, adjacency matrix w
Output: Distance matrix dist
Initialize dist[i][j] ← w[i][j] for all i, j (use ∞ if no edge, 0 if i = j)
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if dist[i][k] + dist[k][j] < dist[i][j] then
                dist[i][j] ← dist[i][k] + dist[k][j]
            end if
        end for
    end for
end for
Return dist
```

3.1 Path Reconstruction

To reconstruct the shortest path, maintain a `next` matrix where $\text{next}[i][j]$ stores the next vertex in the shortest path from i to j. Update it as:

$$\text{if } \text{dist}[i][k] + \text{dist}[k][j] < \text{dist}[i][j], \text{ then } \text{next}[i][j] \leftarrow \text{next}[i][k]$$

The path from i to j is retrieved by following $\text{next}[i][j]$ until reaching j.

4 Complexity Analysis

- **Time Complexity**: $O(n^3)$, due to three nested loops over n vertices.
- **Space Complexity**:
  - $O(n^2)$ for the distance matrix.
  - Additional $O(n^2)$ for the `next` matrix if path reconstruction is needed.
  - Can be optimized to $O(1)$ extra space by overwriting the input matrix (in-place updates).

5 Detecting Negative Cycles

A negative cycle exists if, after running the algorithm, any diagonal element $\text{dist}[i][i] < 0$. This indicates a path from a vertex to itself with negative total weight, implying a negative cycle.

6 Practical Applications

The Floyd-Warshall algorithm is used in:

- **Network Routing**: Computing optimal paths in communication networks.
- **Transitive Closure**: Determining reachability in graphs (Warshall's algorithm variant).
- **GPS Navigation**: Precomputing shortest paths for dense city networks.
- **Social Network Analysis**: Analyzing shortest paths for influence or connectivity.

7 Example

Consider a graph with 4 vertices and the following adjacency matrix:

	1	2	3	4
1	0	3	∞	5
2	∞	0	2	∞
3	1	∞	0	2
4	∞	∞	∞	0

After running Floyd-Warshall, the distance matrix becomes:

	1	2	3	4
1	0	3	5	5
2	3	0	2	4
3	1	4	0	2
4	∞	∞	∞	0

For example, the shortest path from vertex 1 to 3 is 5 (via vertex 4: $1 \to 4 \to 3$).

8 Limitations and Extensions

- **Limitations**:
  - Inefficient for sparse graphs ($O(n^3)$ is costly when $|E| \ll n^2$).
  - Cannot handle negative cycles (requires detection and special handling).
- **Extensions**:
  - **Transitive Closure**: Modify to compute reachability (0/1 matrix).
  - **Parallel Implementation**: Use GPU or distributed systems for large graphs.

- **Approximate Algorithms**: For faster results in massive graphs.

9 Implementation Considerations

- **Numeric Stability**: Use `long long` or `double` for large edge weights to avoid overflow.
- **Sparse Graphs**: For sparse graphs, consider Johnson's algorithm, which combines Dijkstra's and Bellman-Ford for $O(n^2 \log n + n|E|)$.
- **Matrix Optimization**: Use bit-packing or adjacency lists for sparse graphs to reduce memory.

10 References

- GeeksForGeeks: Floyd-Warshall Algorithm
- Wikipedia: Floyd-Warshall Algorithm
- Cormen, T. H., et al., *Introduction to Algorithms*, Chapter on All-Pairs Shortest Paths.