DAA-Notes

16 February 2024

ROLL NO. 220001035,220001041,220001042

1 Floyd-Warshall Algorithm

This Algorithm is used to find the shortest paths between all pair of nodes in a weighted graph. While we can write Bellmann Ford Algorithm for every node in order to obtain the shortest distance between every pair,it has an $O(n^4)$ time complexity and requires numerous repetitions.

Floyd Warshall Algorithm is preferred in order to minimize repetitions and optimize time complexity. It is highly efficient and can handle graphs with both positive and negative edges.

1.1 Assumption

The Floyd-Warshall algorithm assumes that the optimal path between any two vertices does not contain any negative cycles. This assumption is fundamental to the correctness of the algorithm. If there exists a negative cycle, then it is possible to traverse the cycle repeatedly, decreasing the total weight of the path without limit. Therefore, the concept of shortest paths becomes undefined in the presence of negative cycles.

1.2 Working

Consider a graph G with vertices V numbered 1 through n. Further consider a function $W^k[i,j]$ that returns the length of shortest possible path from i to j using vertices from the set 1,2,...,k as intermediate points.

Here $W^k[i,j]$ represents tensor or 3D matrix.

$W^k[i,j]$ = W[i][j][k] , here

i : source

j : destination

k : iteration number

It is not mandatory to have k vertices between i and j.

1.3 Initialization

Here $W^0[i,j]$ represents 2D matrix

$W^0[i,j]$ = w[i,j]

where w[i,j] denotes the weight of the edge from i to j if exists and it is set to infinity otherwise

$W^k[i,j] = \min\{W^k[i,j]\ ,\ W^{k-1}[i,k] + W^{k-1}[k,j]\ \}$

2 Floyd-Warshall Algorithm

We can express the main idea behind the Floyd-Warshall algorithm formally as follows:

For any pair of vertices a and b in the graph, the shortest path between them (denoted by $g[a][b]$) is the minimum of either:

1. The cost of the direct edge from a to b, or
2. The sum of the costs of the paths from a to b through any intermediate vertex k.

Mathematically, this can be written as:

$$g[a][b] = \min(g[a][b], g[a][k] + g[k][b])$$

where k is any vertex in the graph.

This formula captures the essence of the Floyd-Warshall algorithm, which iteratively updates the shortest path distances between all pairs of vertices by considering all possible intermediate vertices.

Example:-

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

For k=1

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

For k=2

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	12	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	3	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-2	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

For k=3

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	12	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	3	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	-1	-2	-1	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-2	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

Listing 1: Pseudocode for Floyd-Warshall Algorithm

```
for each vertex v
    for each vertex u
        dist[v][u] gets Infinity
for each vertex v
    dist[v][v] \gets 0
for each edge (u, v) with weight w
    dist[u][v] gets w
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            dist[i][j] gets min(dist[i][j], dist[i][k] + dist[k][j])
```

The time complexity of the Floyd-Warshall algorithm is $O(n^3)$ because for every vertex taken as an intermediate, we traverse the entire matrix. Thus,

the time complexity becomes $n \times n^2 = O(n^3)$. And we maintain an adjacency matrix of $n \times n$, so space complexity is $O(n^2)$.