

# Floyd-Warshall Algorithm

Ms. Sasmita Kumari Nayak

Computer Science & Engineering

# **Floyd Warshall Algorithm**

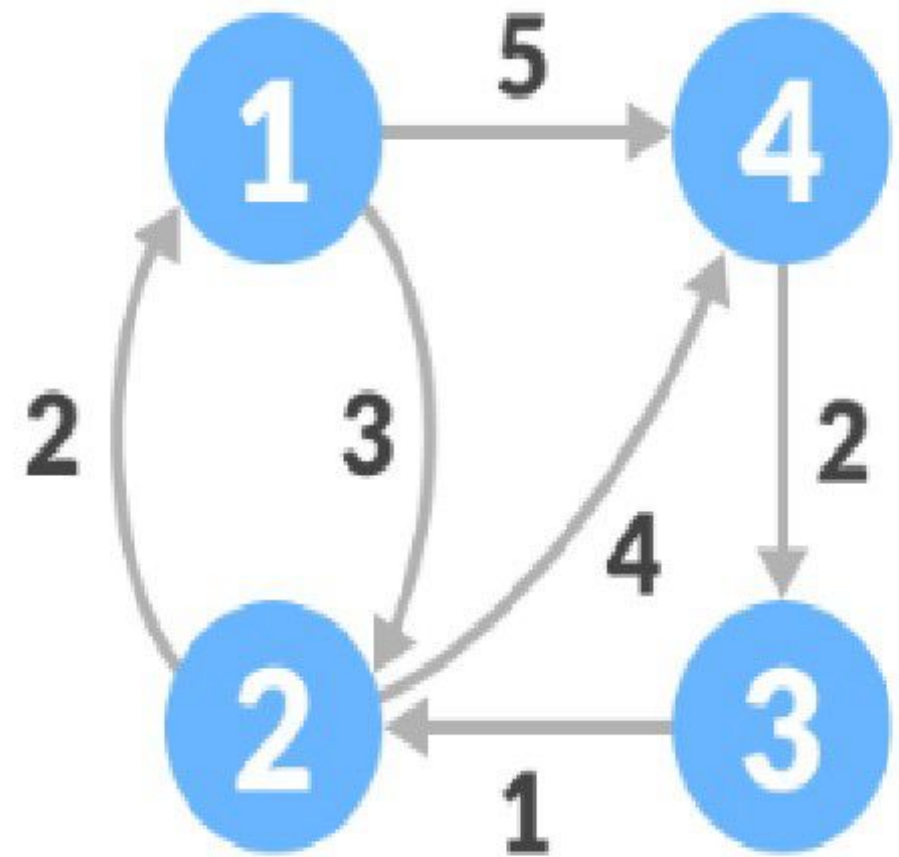
- Floyd Warshall Algorithm is a famous algorithm.
- It is used to solve All Pairs Shortest Path Problem.
- It computes the shortest path between every pair of vertices of the given graph.
- Floyd Warshall Algorithm is an example of dynamic programming approach.

# Advantages

- Floyd Warshall Algorithm has the following main advantages-
  - It is extremely simple.
  - It is easy to implement.

# PRACTICE PROBLEM

- Consider the following directed weighted graph.
- Using Floyd Warshall Algorithm, find the shortest path distance between every pair of vertices.



## Solution

- Follow the steps below to find the shortest path between all the pairs of vertices.

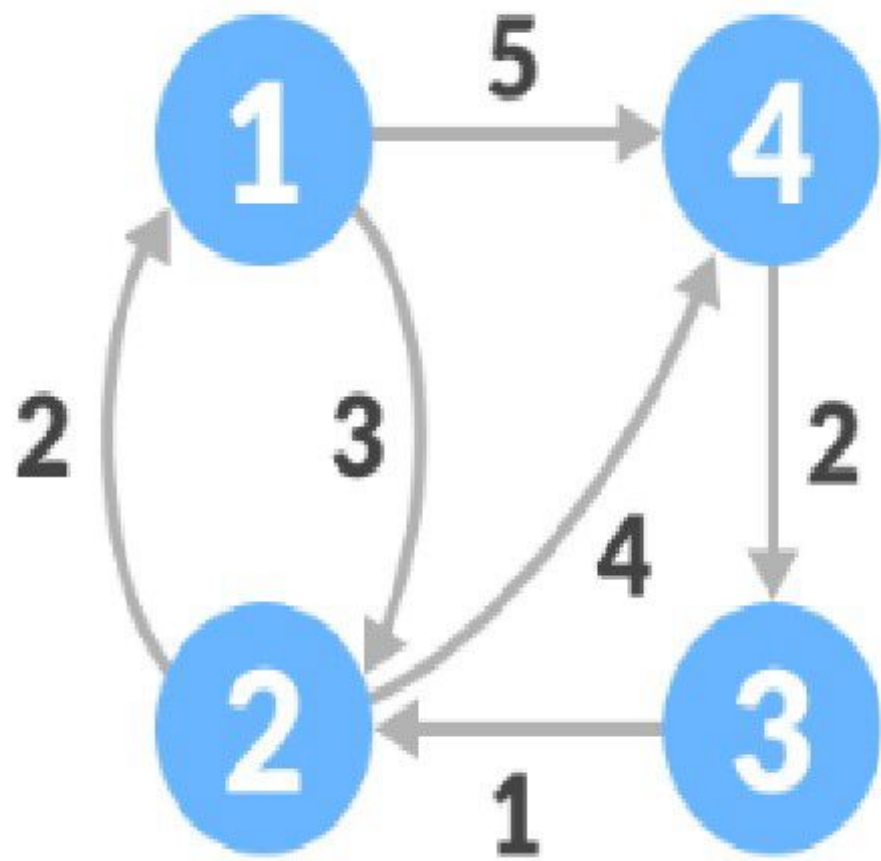
# **Step-01**

- Remove all the self loops and parallel edges (keeping the lowest weight edge) from the graph.
- In the given graph, there are neither self edges nor parallel edges.

## Step-02

- Write the initial distance matrix  $A^0$ .
- It represents the distance between every pair of vertices in the form of given weights.
- For diagonal elements (representing self-loops), distance value = 0.
- For vertices having a direct edge between them, distance value = weight of that edge.
- For vertices having no direct edge between them, distance value =  $\infty$ .

Cont...



$A^0 =$

	1	2	3	4
1	0	3	$\infty$	5
2	2	0	$\infty$	4
3	$\infty$	1	0	$\infty$
4	$\infty$	$\infty$	2	0

## Step-03

- Now, create a matrix  $A^1$  using matrix  $A^0$ . The elements in the first column and the first row as well as diagonal elements are left as they are. The remaining cells are filled in the following way.
- Let  $k$  be the intermediate vertex in the shortest path from source to destination. In this step,  $k$  is the first vertex.

if ( $A[i][j] > A[i][k] + A[k][j]$ ).

then  $A[i][j]$  is filled with ( $A[i][k] + A[k][j]$ )



# Cont...

- That is, if the direct distance from the source to the destination is greater than the path through the vertex  $k$ , then the cell is filled with  $A[i][k] + A[k][j]$ .
- In this step,  $k$  is vertex 1. We calculate the distance from source vertex to destination vertex through this vertex  $k$ .

$$\mathbf{A}^1 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & & \\ \infty & & 0 & \\ \infty & & & 0 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & 9 & 4 \\ \infty & 1 & 0 & 8 \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{matrix}$$

## Cont...

- For example:

For  $A^1[2, 4]$ , the direct distance from vertex 2 to 4 is 4 and the sum of the distance from vertex 2 to 4 through vertex (i.e. from vertex 2 to 1 and from vertex 1 to 4) is 7.

$$A^1[2, 4] = A^0[2, 1] + A^0[1, 4] = 2 + 5 = 7$$

Since  $4 < 7$ ,  $A^0[2, 4]$  is filled with 4.

## Step-04

- In a similar way,  $A^2$  is created using  $A^1$ . The elements in the second column and the second row are left as they are.
- In this step,  $k$  is the second vertex (i.e. vertex 2). The remaining steps are the same as in **step 3**.

$$A^2 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & & \\ 2 & 0 & 9 & 4 \\ & 1 & 0 & \\ & \infty & & 0 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 9 & 5 \\ 2 & 0 & 9 & 4 \\ 3 & 1 & 0 & 5 \\ 4 & \infty & \infty & 2 & 0 \end{bmatrix} \end{matrix}$$

## Step-05

- Similarly,  $A^3$  and  $A^4$  is also created.

$$A^3 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & & \infty & \\ & 0 & 9 & \\ \infty & 1 & 0 & 8 \\ & & 2 & 0 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 9 & 5 \\ 2 & 0 & 9 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{matrix}$$

# Cont...

- $A^4$  gives the shortest path between each pair of vertices.

$$A^4 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & & & 5 \\ & 0 & & 4 \\ & & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 7 & 5 \\ 2 & 0 & 6 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{matrix}$$

Algorithm

```

//It represents the distance between every pair of vertices as
given.
//n is the number of vertices present in the graph

Create a n x n matrix

For each cell (i, j) in D do-

    // for all diagonal elements, value = 0

    if i == j

        D[i][j] = 0

    // If there exists a direct edge between the vertices,
    //value = weight of edge

    if (i, j) is an edge in E

        D[i][j] = weight (i,j)

    // If there is no direct edge between the vertices, value =  $\infty$ 

    Else

        D[i][j] = infinity

```



```
// Find the distance for all pair shortest path  
  
for k from 1 to n  
  
    for i from 1 to n  
  
        for j from 1 to n  
  
            if  $D[i][j] > D[i][k] + D[k][j]$   
                 $D[i][j] = D[i][k] + D[k][j]$ 
```



# Time complexity

- Floyd Warshall Algorithm consists of three loops over all the nodes.
- The inner most loop consists of only constant complexity operations.
- So, the time complexity of the Floyd-Warshall algorithm is  $O(n^3)$ .
- Here,  $n$  is the number of nodes in the given graph.

# **Applications of Floyd Warshall** **Algorithm**

- To find the shortest path in a directed graph
- To find the transitive closure of directed graphs
- To find the Inversion of real matrices
- For testing whether an undirected graph is bipartite

# Assignment

- Using Floyd Warshall Algorithm, find the shortest path distance between every pair of vertices.

