**def floyd\_warshall(graph):**

**"""**

**Function to implement Floyd-Warshall algorithm for finding**

**the shortest paths between all pairs of nodes in a graph.**

**:param graph: 2D matrix representing the graph where graph[i][j] is the distance from node i to node j**

**:return: 2D matrix of shortest paths between all pairs of nodes**

**"""**

**# Get the number of vertices (nodes) in the graph**

**n = len(graph)**

**# Initialize the distance matrix as a copy of the input graph**

**dist = [[graph[i][j] for j in range(n)] for i in range(n)]**

**# Main loop of Floyd-Warshall: consider each node k as an intermediate node**

**for k in range(n):**

**# Consider every pair of nodes (i, j)**

**for i in range(n):**

**for j in range(n):**

**# If going through node k offers a shorter path from i to j, update the distance**

**dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])**

**# Return the distance matrix containing the shortest path distances between all pairs**

**return dist**

**# Example usage**

**if \_\_name\_\_ == "\_\_main\_\_":**

**# Define the graph as a 2D list (adjacency matrix)**

**# graph[i][j] represents the distance from node i to node j**

**# Use float('inf') for nodes that are not directly connected**

**INF = float('inf')**

**graph = [**

**[0, 3, INF, 7],**

**[8, 0, 2, INF],**

**[5, INF, 0, 1],**

**[2, INF, INF, 0]**

**]**

**# Call the Floyd-Warshall algorithm**

**shortest\_paths = floyd\_warshall(graph)**

**# Print the resulting shortest path matrix**

**print("Shortest distance matrix between all pairs of nodes:")**

**for row in shortest\_paths:**

**print(row)**