**THEORY**

**📌 Graph Representation**

A **graph** is a collection of **nodes (vertices)** and **edges**. In this program:

* Each **city** (represented by an airport code) is a **node**.
* A **distance** (non-zero) between two cities indicates a **connection (edge)**.
* The **graph is undirected**: if there is an edge from A to B, there is also an edge from B to A.

**🔹 Adjacency Matrix**

* A 2D array where adj\_mat[i][j] = 1 (or a distance value) if there is an edge between city i and city j.
* Symmetric for undirected graphs.

**📌 Depth-First Search (DFS)**

DFS is a **recursive** algorithm to explore a graph by **going as deep as possible** before backtracking.

* It uses **recursion** and a **visited[] array** to avoid revisiting nodes.
* DFS is useful for tasks like **pathfinding**, **cycle detection**, or **tree/graph traversal**.

**📌 Breadth-First Search (BFS)**

BFS is an **iterative** algorithm that explores the graph **level by level** using a **queue**.

* It starts from a node and explores all its neighbors before moving to the next level.
* BFS is useful in **finding shortest paths** and **minimum distance** in unweighted graphs.

**🔷 ALGORITHMS**

**🔧 1. DFS (Depth-First Search)**

text

CopyEdit

Input: Starting vertex u, number of cities n, city name array arr[]

Output: DFS traversal order

Algorithm DFS(u):

1. Mark u as visited

2. Print arr[u]

3. For each vertex i from 0 to n-1:

a. If adj\_mat[u][i] == 1 and visited[i] == 0:

- Recursively call DFS(i)

**🔧 2. BFS (Breadth-First Search)**

text

CopyEdit

Input: Starting vertex u, number of cities n, city name array arr[]

Output: BFS traversal order

Algorithm BFS(u):

1. Initialize a queue and visited[] array

2. Enqueue u, mark visited[u] = true

3. While queue is not empty:

a. Dequeue front vertex v

b. Print arr[v]

c. For each vertex i from 0 to n-1:

- If adj\_mat[v][i] == 1 and visited[i] == false:

- Enqueue i and mark visited[i] = true

**🔧 3. Adjacency Matrix Initialization and Input**

text

CopyEdit

1. Read number of cities n

2. Input city names into arr[]

3. For each pair (i, j) with i < j:

- Input distance

- Set adj\_mat[i][j] = adj\_mat[j][i] = distance

