## **(a). Graph Representation**

## Chosen Data Structure: **An Adjacency Matrix**

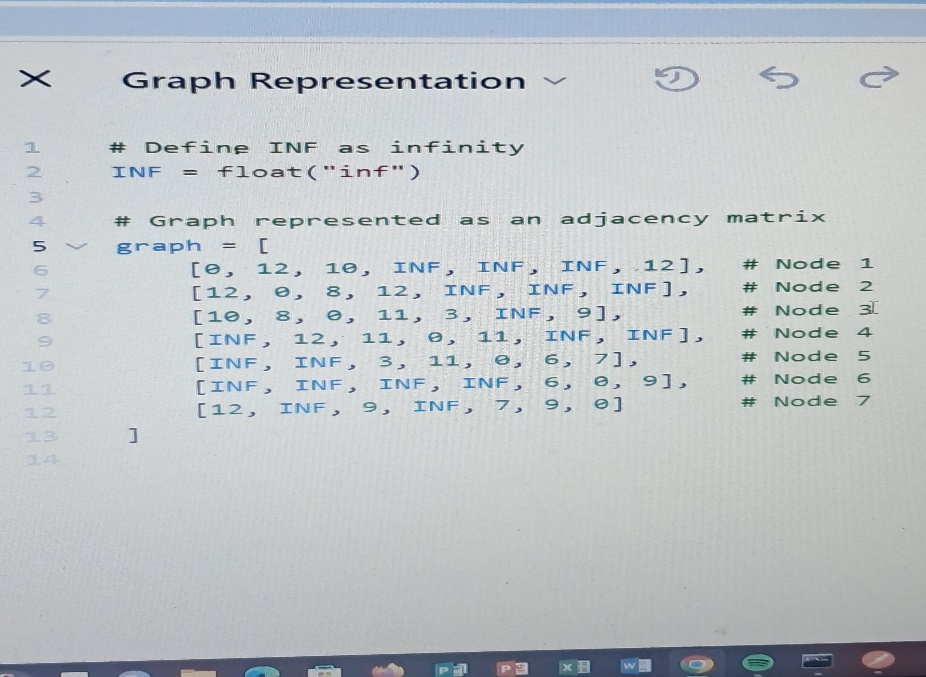
An **Adjacency Matrix** is a **2D array** used to represent a graph, where rows and columns correspond to cities (nodes). Each entry Matrix[i][j] stores the weight (distance) between city i and city j, with 0 for the same city (i=j), the actual distance if a direct route exists, and ∞ (or a large value) if no direct connection exists.

#### **2. Adjacency Matrix for the Given Graph** Here’s how the matrix looks:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **1** | 0 | 12 | 10 | ∞ | ∞ | ∞ | 12 |
| **2** | 12 | 0 | 8 | 12 | ∞ | ∞ | ∞ |
| **3** | 10 | 8 | 0 | 11 | 3 | ∞ | 9 |
| **4** | ∞ | 12 | 11 | 0 | 11 | ∞ | ∞ |
| **5** | ∞ | ∞ | 3 | 11 | 0 | 10 | 7 |
| **6** | ∞ | ∞ | ∞ | ∞ | 10 | 0 | 9 |
| **7** | 12 | ∞ | 9 | ∞ | 7 | 9 | 0 |

* **Diagonal values are 0** (distance from a city to itself).
* **"∞" represents no direct route between cities.**

#### **Python Code for Adjacency Matrix Representation**

****Here's how you can implement this in **Python**:

**Justification of Adjacency Matrix**

The **Adjacency Matrix** is chosen because it allows **O(1) time** retrieval of distances using **graph[i][j]**, ensuring fast lookups for the TSP problem. It requires **O(N²) memory**, which is efficient for small, fully connected graphs. Additionally, its **simple 2D list structure** makes it easy to implement and understand.

(b) **Problem Setup**

In the context of the Traveling Salesman Problem (TSP) using an adjacency matrix, the problem setup can be clearly restated as follows:

**TSP Objective:**

The objective is to find the shortest possible route that allows a salesman to:

1. **Visit each city exactly once**: The salesman must visit each city in the given set of cities only one time.
2. **Return to the starting city**: After visiting all the cities, the salesman must return to the city where the journey began.
3. **Minimize total travel distance**: The total distance traveled during the entire trip should be as small as possible.

**Assumptions Made:**

1. **Symmetric distances**: It is assumed that the distance between any two cities is the same in both directions (i.e., traveling from city1 to city2 has the same cost as traveling from city2 to city1).
2. **Non-negative distances**: All the distances between cities are assumed to be non-negative. In practical terms, this means the distance between any two cities cannot be negative, and self-loops (distance from a city to itself) are assumed to be zero.
3. **Complete graph**: The adjacency matrix is assumed to represent a complete graph, meaning there is a direct path (a non-infinite distance) between every pair of cities.
4. **Matrix representation**: The adjacency matrix is a square matrix where each element A[i][j] represents the distance between city i and city j. If no direct path exists between two cities, a very large value (often infinity) is used in the matrix to indicate the lack of a direct connection.