Draft Version: 0.0

**MAKERERE** **UNIVERSITY**

****

**AN ACADEMIC ISSSUE TRACKING SYSTEM (AITS):**

Case Study: MAKERERE UNIVERSITY

By:

**GROUP\_S**

COLLEGE OF COMPUTING AND INFORMATION SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLY

DEPARTMENT OF COMPUTER SCIENCE

BACHELOR OF SCIENCE IN COMPUTER SCIENCE (Year One)

(CSC: 1204). Data Structures & Algorithms Assignment of **Travel Salesman Problem (TSP) Using Classical & SOM-Based Methods.**

**Chosen Approach: Adjacency Matrix**

**semester II** Academic Year:

**2024:**

**Due date: 25TH March 2025**

# Assignment Team Members

**Document Change Control**

|  |
| --- |
| Name REG.NO Role/Responsibility |

|  |  |  |
| --- | --- | --- |
| 1. Seanice Nabasirye   (0770862549) | 24/U/22871/PS | Documentation |
| 1. Asiimwe Isaac   (0770563394) | 24/U/03765/PS | Front-End Developer/Team leader |
| 1. Kiyingi Solomon Jessy   (0767594340) | 24/U/25920/EVE | Back-End Development |
| 1. Jonah Akandwanaho   (0756348528) | 24/U/03053/PS | Front-End Development |
| 1. Keith Paul Kato   (0759021371) | 24/U/26593/EVE | Back-End Development |
|  |  |  |

# **Task 1: Representation & Data Structures:**

The **(TSP)** can be effectively represented using an **Adjacent Matrix Approach,** a data structure most appropriate for storing the graph representation of the city and the distances. It uses a 2D-Array matrix that can also be used for a dense graph where all nodes are connected as illustrated by the graph representation of the city and distances in the Problem Statement. **STRUCTURE OF ADJACENT MATRIX APPROACH**

The graph representation of cities and distances has 7 cities therefore a 7\*7 matrix is used. In addition to that if there’s no direct route between any city, we set matrix **[i][j]** = **∞** which means “Infinite distance” or “no connection between the cities” where i and j are cities. Furthermore, since the graph is undirected (there can be a route to and from, two cities i.e., from city 1 to city 2 & 2 to 1), the matrix is symmetric (**matrix[i][j]** = **matrix[j][i]**). With diagonal elements set to **zero (0)**. This representation allows efficient loom uo **REASONS FOR CHOOSING ADJACENT MATRIX APPROACH.**

1. The adjacent matrix is good for dense graphs, it efficiently stores all connections.
2. Checking if an edge exists between 2 nodes is 0(1). 0(1) means that, the operation takes the same amount of time regardless of how many elements are in the data structure.
3. It is easy to implement using a 2D array.

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