### ÇANKAYA UNIVERSITY

### SOFTWARE ENGINEERING DEPARTMENT

### SOFTWARE PROJECT III



|  |  |
| --- | --- |
| **Name Surname** | **Ferhan Burak Özkan** |
| **Identity Number** | **202028018** |
| **Course** | **SENG 383** |
| **Experiment** | **Transportation Network** |
| **E-mail** | **c2028018@student.cankaya.edu.tr** |

# **PROBLEM STATEMENT**

* Goals of the Programming Assignment: The goal of this programming assignment is to implement a solution for the Traveler Problem using undirected graphs. The problem involves finding paths between cities using different transportation types (Highway, Airway, Railway). Three types of queries are provided to inquire about possible travel paths.
* Normal Inputs to the Program: The program takes input matrices representing the adjacency of cities for three transportation types.

Query file (query.inp) is used to obtain information about specific travel scenarios.

* Output of the Program: The program should generate paths based on the queries, considering transportation types and city connections.
* Error Handling: Proper validation should be implemented to handle cases where queries or input data are incorrect.

**DESIGN**

Design Decisions

* Combined all transportation types into one graph with different colors representing each type.
* Implemented a dynamic matrix to handle different city sizes.

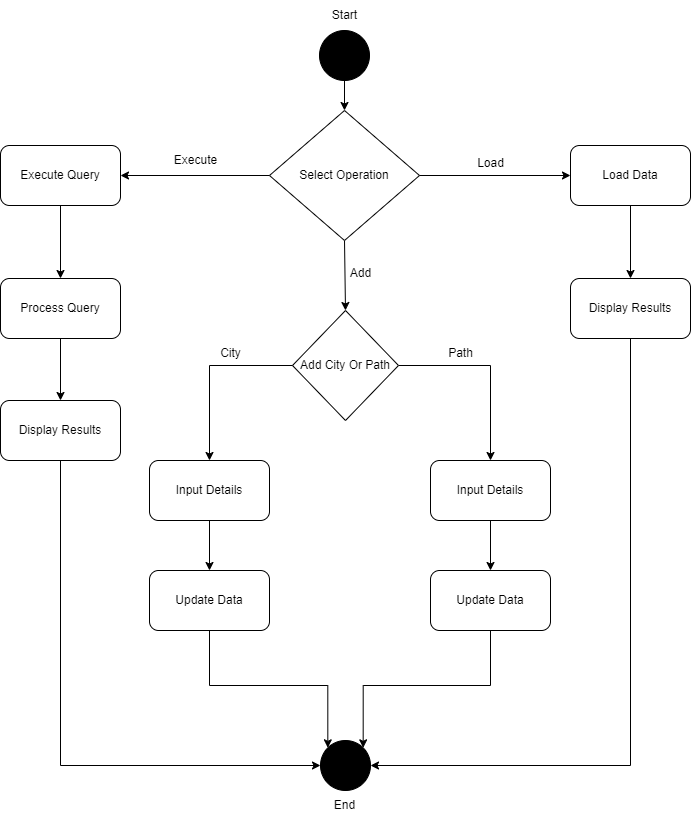
Data Structures and Algorithms

* Used an adjacency matrix to represent the graph.
* Employed depth-first search (DFS) algorithm to find paths.

Pros/Cons of Choices

* Pros: Simple representation, easy to implement.
* Cons: May not be optimal for very large graphs

**IMPLEMENTATION DETAILS**



Adaptation of Code

* Started with basic functions for reading input matrices, creating a graph, and applying depth-first search (DFS).
* Extended the code to handle three types of transportation matrices.
* Adapted the DFS algorithm to consider transportation types and constraints specified in the queries.
* Added functions to process different query types (Q1, Q2, Q3) and generate paths accordingly.
* Implemented functions to display results based on the query type.
* For bonus part 1, modified the path generation to find paths without considering the given order of transportation types.
* For bonus part 2, added functions to handle commands for adding a new city and a new path to the graph.

Development Timeline

* Research and planning - Week 1
* Implementation of graph representation - Week 2
* Query processing implementation - Week 3
* Testing and debugging - Week 4

**TESTING NOTES**

* Testing Process

Tested the program with various input matrices and query scenarios.

* Normal Inputs

Input matrices representing city connections for three transportation types.

Query file with different scenarios.

* Special Cases

Tested with queries requiring multiple transportation types and paths with specific constraints.

* Results

The program produced correct paths for the given queries.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazı tipi, ekran görüntüsü, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**COMMENTS**

* Overall Result

The programming project was a success; the implemented solution met the specified requirements.

* Success Criteria

The program correctly handled various queries and generated paths.

* What Would You Do Differently?

Consider optimizing the graph representation for larger datasets.

Enhance error handling for more robustness.