PFL - Project 1

T16_G07

Maria Eduarda Pacheco Mendes Araújo - up202004473 --> 50%

- Helped more with the last 2 functions as they were the hardest ones.

Miguel Ângelo Pacheco Valente - up201704608 --> 50%

- Coded the first few functions as they were small and easy and worked together for the last 2 functions.

We worked together throughout the project but more intensely in the last 2 functions.

General Description

In this project we're asked to define and use appropriate data types for a graph representing a country, composed of a set of interconnected cities, also testing the quality and efficiency of the code.

---

shortestPath function

The **shortestPath** function is an adaptation of ***Dijkstra's algorithm*** to find the shortest path between two cities (*start* and *end*) in a *RoadMap*.

Data Structures:

- **City** --> *String* --> City name
- **Path** --> *List* --> Represents a path
- **Distance** --> *Int* --> Distance between cities
- **RoadMap** --> *List of tuples* --> Represents direct connections between cities with the respective distance

Helper Functions:

- **cities** --> Extracts all unique cities from *RoadMap*
- **areAdjacent** and **distance** --> Checks adjacency between two cities and retrieves the distance
- **adjacent** --> Returns all cities directly connected to a specific city along with their respective distances
- **toAdjList** --> Converts the *RoadMap* into an adjacency list, which facilitates Dijkstra's algorithm
- **lookupAdjacent** and **lookupDistance** --> Helper functions to get direct neighbours and distances between cities from the adjacency list
- **pathDistanceFrom** --> Helper function to calculate the distance of a path from the adjacency list

---

travelSales function

The **travelSales** function provides a solution to the **Traveling Salesman Problem** (*TSP*) by using ***greedy algorithm***.

TSP aims to find the shortest possible route that visits every city exactly once and returns to the starting city.

Data Structures:

- **City** --> *String* --> City name
- **Path** --> *List* --> Represents a path
- **Distance** --> *Int* --> Distance between cities
- **RoadMap** --> *List of tuples* --> Represents direct connections between cities with the respective distance

Helper Functions:

- **findGreedyPath** --> Finds a greedy path for the TSP recursively
- **pathDistance** --> Returns the sum of all individual distances in a path between two cities in a Just value, if all the consecutive pairs of cities are directly connected by roads
- **toAdjList** --> Converts the *RoadMap* into an adjacency list format making it easier to access neighbouring cities and their distances
- **lookupAdjacent** --> Finds all cities directly connected to a given city in the adjacency list
- **lookupDistance** --> Retrieves the distance between two cities from the adjacency list