U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Programação Funcional e em Lógica

T15_G03

David Carvalho – up202208654@up.pt

Diogo Vieira – up202208723@up.pt

Índice

Group Members and Contributions

- **Diogo da Silva Vieira** - 50%
  - **Email:** up202208723@up.pt
  - **Tasks:** Implemented the cities, areAdjacent, distance, and pathDistance functions. Worked on the travelSales function and wrote the auxiliary functions tsp and bestPath.

- **David Gustavo Campos Carvalho** - 50%
  - **Email:** up202208654@up.pt
  - **Tasks:** Implemented the rome, isStronglyConnected, dfs, and dfsPaths functions. Worked on the shortestPath function and wrote the auxiliary function adjacent.

Shortest Path Function Implementation

Explanation

The shortestPath function finds all the shortest paths between two cities in a given road map. It returns a list of paths that share the minimum distance. This approach ensures that all possible shortest routes are considered.

Algorithm, Data Structures and Helper Functions

- **Data Structures**:
    - **RoadMap**: A list of tuples, each representing a road between two cities with an associated distance.
    - **Path**: A sequence of cities representing a route.
    - **Distance**: An integer indicating the total length of a path.
- **Algorithm**:
    - **DFS with Path Tracking**: The dfsPaths function employs a depth-first search to find all paths between two cities. It recursively explores each route while avoiding cycles by maintaining a list of visited cities.
    - **Path and Distance Collection**: Each path and its corresponding distance are stored and evaluated. The shortest paths are filtered by comparing their distances to the minimum found.
- **Helper Functions**:
    - **dfs**: Ensures all cities can be reached from the starting point.

Justification

Depth-first search was selected for its capability to explore all potential paths exhaustively. The recursive design of dfsPaths ensures thorough path exploration. Lists were utilized for path tracking due to their simplicity in recursive functions.

Travel Sales Function Implementation

Explanation

The travelSales function aims to solve the TSP for a given road map by finding a path that visits each city exactly once and returns to the starting point. The function returns the most optimal path found.

Algorithm, Data Structures and Helper Functions

- **Data Structures**:
  - **RoadMap**: A list of tuples indicating roads between cities and their distances.
  - **Path**: Represents a sequence of visited cities.
  - **Distance**: Represents the path's cumulative distance.
- **Algorithm**:
  - **Recursive Path Generation (TSP)**: The tsp function iteratively constructs paths by adding unvisited cities, calculating distances along the way.
  - **Path Selection**: bestPath evaluates all complete paths and selects the one with the lowest total distance.
- **Helper Functions**:
  - **tps**: This function finds all possible paths for the Traveling Salesman Problem (TSP).
  - **calculateTotalDistance**: Computes the total length of a path by summing the distances between consecutive cities.

Justification

A recursive approach was employed to explore possible paths in the TSP, providing a straightforward mechanism for calculating routes and handling returns to the starting city. This method, while not optimal for large inputs, is appropriate for moderate-sized datasets where full exploration is feasible.

Final Note

The functions incorporate comprehensive path tracking and distance calculations, ensuring that solutions prioritize correctness. Helper functions, such as dfs and calculateTotalDistance, aid in modular and readable code structure, also there are comments to facilitate understanding and maintenance. The solutions prioritize clarity and correctness, with efficiency considerations addressed where possible.