T14_G03

Group Members

- Rafael Filipe Barbosa da Costa up202205013 (Contribution: 50%)
- Miguel Wojciech de Vasconcelos Charchalis up201506074 (Contribution: 50%)

ShortestPath Function

The shortestPath function uses Dijkstra's algorithm to find all shortest paths between two cities in a roadmap (RoadMap). This function is designed to return a list of shortest paths, which means it not only finds the shortest distance between the start and end cities but also considers multiple paths if they are of the same minimal distance.

Base Case Check:

If start and end are the same, it returns [[start]], as there’s no path required to travel between the same cities. Otherwise, it initializes the Dijkstra’s process by calling the helper function dijkstra

Dijkstra Helper Function:

The dijkstra function processes cities from the priority queue, always choosing the one with the shortest distance first.

If it reaches end, it adds the path to shortestPaths if it’s the shortest seen or has an equal distance to others already stored.

If currentCity isn’t end, it marks it as visited, expands to unvisited neighbors, creates new paths, and re-sorts the queue by distance.

Data Structures:

- The priority queue `[(City, Path, Distance)]` ensures cities are processed by shortest cumulative distance.
- Visited list `[City]` prevents re-processing cities, reducing redundancy.
- ShortestPaths list `[(Path, Distance)]` stores all paths with minimal distance to end, allowing the function to return multiple shortest paths.

Algorithm Justification

Dijkstra’s algorithm was chosen because it’s efficient in finding the shortest path in weighted graphs without negative weights. By maintaining a priority queue and only expanding nodes with the shortest cumulative distance, the algorithm ensures that it finds the shortest path in an optimal and computationally feasible manner. Additionally, the adaptation to collect multiple paths (if they have equal shortest distances) aligns well with Dijkstra’s framework by only requiring small modifications to track alternative shortest paths when encountered.

TravelSales Function

Two datatypes were defined:

- TspCoord that keeps track of the current city and the cities that have yet to be visited
- TspEntry that represents the best solution for a given TspCoord keeping track of the total distance of the path and the path itself

The travelSales function tries to solve the tsp problem from every start/end city and returns the shortest one. It uses TspCoord to track the current and remaining cities and TspEntry to store the best solution for each TspCoord so they can be used later if needed.

For each starting city, it is removed from the list of remaining cities and compTsp is called. compTsp recursively explores possible paths checking for remaining cities and calculating the distances to potencial next cities. It uses the compareTspEntry function to keep track of the shortest solution for each state.

After all paths are calculated the path with the least distance is returned. If no path is found, an empty list is returned