Design Document

Background

A delivery company which processes orders in the order of the soonest deadline using a max-heap, then finds customer information using Red-Black Tree, and finally finds the shortest path to the delivery destination using Dijkstra's algorithm.

This program will use Java as the programming language.

Functionalities:

(I/O: Read in the provided input and save the information in data structure.)

1. Identify the order with soonest delivery deadline among all the orders.
2. Find the order information including name, address, delivery deadline, and order ID with a given order ID.
3. Find the closest path to the desired destination (Address ID).

Data structures and algorithms:

- Heap (Functionality 1)

This algorithm is implemented in the Heap class. Given a list of customer orders, heap sort will be implemented to sort these orders in an ascending order of the soonest deadline, in functionality 1, the order information with soonest delivery date can be obtained by accessing the first element (order) in the sorted list.

Having a list sorted in advance will make the further accesses simpler and faster, theoretically, heap sort will take a time complexity of O(nLogn) to complete, and the access to the order of soonest deadline can be done in a O(1) time complexity, and this will enhance the efficiency of accessing the element.

- Red-Black Tree (Functionality 2)

This data structure is implemented in the RBTree class. With the given list of orders as above, the data will be inserted into a Red-Black Tree with assist of relevant methods within the same class, the tree formulation is based on the order IDs, hence the functionality 2 can access the order information from the Red-Black Tree according to these IDs.

The properties of Red-Black Tree making it a balanced tree and the number of rotations for an insertion is lower than AVL tree in average. Accessing an element in a balanced tree would guarantee the stability of time taken, hence I have chosen Red-Black Tree to be the data structure of storing order information.

- Dijkstra Algorithm (Functionality 3)

Dijkstra will be used to find the shortest path between two locations, to make the algorithm more intuitive, I decided to denote every location on the map by an address ID, throughout the program, the address of each customer is also represented by an address ID. In this algorithm, the shortest path to an address from the delivery company with consideration of connections and distances between different locations.

This algorithm will suit the above situation as it is a path finding approach from a fixed point, which would be the delivery company in this case. Delivery company will be able to lower the time and cost of delivery if a shortest path can be found.

Assumptions:

There are 4 inputs to be given by the user in this program, brief descriptions are shown below:

1. `listOfOrders`: A list of Orders that will be sorted using heap sort and inserted

in the Red-Black Tree. (Functionality 1, 2)

Assumptions: The orders are expected to be in the form of the Order class.

2. `map`: a 2D array providing the connections and their distances between address IDs. A full map with all possible connections between all IDs needs to be provided. (Functionality 3)

   Assumptions: map[x][y]==0 should be satisfied when x == y. map is a square 2D array.

3. `destinationID`: A desired address ID (int) that will help returning the shortest path to this point from where the delivery company is located. (Functionality 3)

   Assumptions: The address ID should be valid from the map above.

4. `findCustomerByID`: This ID will return its corresponding order information. (Functionality 2)

   Assumptions: The order ID should be valid from the input orders above.

All input information will be certain and will not change over the runtime except it may be marked off when its calculation is done. Some examples and class specifications can be found in the README.md file in the assignment package.

Difficulties

This program connects its functionalities with real life situations aiming at assisting them on decision makings. However, In real life, there are likely to be some exceptions where the program isn’t feasible enough to help with making best decisions, for example, the cost of deliveries does not only rely on the distance but also some traffic factors and more considerations.