Dial-a-ride Report

DataStructures:

Used hashing and arrayLists as datastructues for easy access and storage of node to node distances, vehicles, requests data. Below are the classes used in the project to store the data.

// instance of this class holds all the input data and processed data for easy access.

```
public class RideInputData
{

	// # of locations in the city
	private int noOfLocations;

	// # of vehicles in the city
	private int noOfVehicles;

	//max#  ofpassengers in each vehicle
	private int vehicleCapacity;

	// # of requests
	private int noOfRequests;


	 // to store each node to other nodes given distances, shortestdistances
	private Map<Integer, List<NodeToNodeData>> locDistances;

	// to store vehicle related data.
	private Map<Integer, Vehicle> vehicleData;

     // to store vehicle requests related data.
	private Map<Integer, List<Request>> requests;

     //to track the vehicles present at particular location
	private Map<Integer, List<Integer>> vehiclesTracker;

}


public class NodeToNodeData
{

	// it represents given input distance between src node to dest node
	private int inputDistance;

	// it holds shortest distance calculated between src node to destination
	// node
	private int shortDistance;

	// it holds the previous node of dest node along shortest path
	private int previousNode;
}
```

```java
public class Vehicle
{

	// storing unique identification # for each vehicle
	private int vehicleNo;

	// location where vehicle is available.
	private int locationPoint;

	// to keep totalfare earned by the vechile of all the requests it served
	private int fareEarned;

	// vehicle started time when the request is accepted
	private int strtTime;

	// vehilce ended time when the request is served
	private int endTime;

	// vehicle location once after the request is completetly served
	private int destPt;

	// path along which vehicle is passing through
	private String travelPath;

	// vehicle location-time map representing the time at which vehicle
	// passes through particular location
	private Map<Integer, List<Integer>> traversePoint;

	// to store the list of requests it served
	private List<Request> requests;
}


public class Request
{
	// request lower interval pickup time
	private int low_pktime;

	// request upper interval pickup time
	private int up_pktime;

	// request pickup point
	private int src_point;

	// request drop point
	private int dest_point;

	// requested passenger passing through the path
	private String path;

	// vehicle alloted time for this request
	private int req_alloc_time;
}
```

Algorithms:

First implemented Dijkstra's algorithm to find the shorted path between one src node to all nodes, then applied the same for all the nodes keeping one node as src node in each iteration for finding shortest paths between all pairs of nodes but it is taking very long time in running (approx in hrs time) because of heavy calculations involved in O(n3).

4.2.1 section A Link State Routing Algorithm(Dijkstra's algorithm) from from Kurose and Ross Computer Networking a Top down Approach .

then switched to implementation of Floyd-Warshall algorithm to find the shortest path between all pairs of given nodes where only one comparison and one assignement operations is required in O(n3) running with in a second time.

25.2 The Floyd-Warshall algorithm from Introduction to Algorithms 3rd additon Cormen

By R PrashanthKumar Reddy
MT2013118