DIAL A RIDE PROBLEM

PROBLEM SCENARIO:

Given here are different locations of a city. Then we have the distance from one location to every other location across the city. There are many requests from the passengers present at these locations at different point of time for a specific period. These passengers are waiting for a cab to take them their destination location. Also there are cabs which have the responsibility of carrying passenger to their respective destination. The cab can take more than one passenger and then fullfil their request simultaneously.

AIM :

The main aim of the problem is to schedule the cabs efficiently and serve as many as request as possible and generate large revenue.

Constraints:

1. At any given instant the number of passenger in a taxi is at most the capacity of the taxi.
2. Passengers have to be picked up from their source location within a defined period of time.
3. Their is no restriction over the time at which passengers are dropped to their destination location.
4. The passenger pays according to the shortest distance between the source and destination location.

Assumptions:

1. Each of the taxi is considered as an independent entity. The updation of information in one taxi is independent of the other.
2. The speed of taxi is 1km/2minutes.
3. All the time is calculated and given in terms of minutes.
4. All the taxi time is 0 in the beginning indicating mid night.

Repesentation of Input Data:

The distance between two locations are kept in a adjacency matrix, here each of the location of the city is considered as one node and the distance in between the locations is edge weight of the two city location nodes in the graph.

There are various attributes like capacity of a cab, total number of request, total cabs in the city, total number of locations in the city are stored within different variables.

There are some structures that are used to represent the data given as a input:

Taxi Structure:

The taxi structure consists of information as

1. Taxi id → To uniquely identify a taxi.
2. Taxi location → Current location of the taxi.
3. Taxi Time → Time at which a request is processed
4. Capacity → Current capacity of the taxi
5. Revenue → To calculate the revenue of the particular taxi.

Request Structure:

The request structure consists of following information :

1. Request id → To uniquely identify a request.
2. Source → The source location.
3. Destination → The destination location of the request.
4. Waiting start time → the time at which the request starts waiting at its source location
5. Waiting end time → the time at which the request ends waiting at its source location.

output Structure:

The output structure consists of following information :

1. Taxi id: The taxi which processed a certain request.
2. Request: The request processed by a particular taxi.
3. Revenue : The revenue of a particular request.
4. Source → The source location.
5. Destination → The destination location of the request.

Distance array:

2 Dimensional arrays is used for representing all the locations and the distance between them,

Approach:

Main approach used here is to take a request and then find the taxi which satisfies the time constraints specified by the passenger and also is not full along meanwhile we keep on updating the total revenue . Also while adding passenger to the taxi check we also update the revenue, time, capacity and location of the taxi.

Dropping while Picking:

Also while adding passenger to the taxi check if there is any request already present in the taxi whose destination location is same as current request source location then drop that passenger to that location and also update the information.

Picking while Dropping :

If the number of the passengers in the cab is equal to the capacity of the cab then sort all the request on the basis of there shortest distance from the current cab position.

Request with shortest distance is chosen and dropped to its destination while updating current cab position to its destination we check from the remaining requests if any request could satisfy the constraints of the taxi then we allocate it to the taxi, information about the current location of the cab, current time of the cab, total revenue associated with the cab is updated.

Algorithm:

1. Take input from the file the variables like capacity of cab, number of locations in the city, number of requests, number of cabs.
2. Take input from the file the input containing distance between the different locations in the city and store it in a 2 dimensional array.
3. Apply floyd warshall algorithm to calculate the minimum distance in between all the locations in the city.
4. Create structure for both the taxi and the request.
5. The taxi structure have taxi id (for unique identification of the taxi), taxi time , current location of the taxi (where the location and the time keeps on updating along with the movement).
6. Request structure have request id(unique identification of each passenger), start time of waiting, end time of waiting, source location , destination location.
7. Sort the request on the basis of start time of waiting of the request.
8. For each of the request

   a. sort the cabs according to the shortest distance in non decreasing order between the cab and the source location of the request.

   b. for each taxi we check

b1. if the taxi is able to reach to passenger within its specified time limit , if yes then update the current time and location associated with the taxi also update the revenue and add the request to the cab.

Meanwhile if there is any request in the taxi whose destination is the source location of the current location drop it to its destination decrease the count of total number passenger in the taxi.

otherwise Repeat step 8 for other taxi

b2. If at any time the number of passengers exceeds the capacity of taxi then for that taxi keep on dropping the passenger (by dropping the nearest passenger first and then again we check the nearest and so on) to their respective destination until no passenger is left in the taxi.

Along with it also check for all the request not processed yet that if there is a a request which satisfies the taxi time and location constarint. If it does then allocate the request to taxi and update all the information.

9. If the request is not processed by any taxi leave that request and repeat step 8 for other taxi .

Optimization used:

1. Using the concept of Drop While Pick. This will help to drop the passenger while picking another request from the specified location and decrease overhead of coming back to the location to drop the same passenger.
2. Dropping in ascending order of distance from current location of taxi: Once the capacity of taxi is full then get the minimum distance from source to destination of any of the request is calculated and then after dropping the passenger we apply the same algo to get the next nearest distance and so on.
3. Wait for the request if taxi has reached before strating wait time of the request : In order to increase possibity of the gaining revenue if a taxi reaches a location before time then wait for the request to arrive.
4. Check for the nearest taxi first and if it satisfies the request constraints laid by the request allocate request to the taxi: This is done to allocate the passenger the taxi at minimum location from its source location.
5. While dropping the passenger to their destination location identify if there is any request which is present within the destination location of the request to be dropped. If there is such request and it is waiting during the time cab is dropping passenger then take the request and update the information about the request within the structure.

REFERENCES:

1. www.stackoverflow.com
2. Prashant Ruwali
3. Devesh Singh Rawat
4. 4. Ajay Tiwari
5. Algorithms(CLR)