CS 501 Data Structures and Algorithm

Mini Project
Project: Dial A Ride

Submitted by: **Joydeep (MT2013062)**
Guided by: **Prof.Dr. Muralidhara V N**

Problem Definition:

To write a program to schedule the N vehicles in such a way as to maximize the revenue. You can decide to reject a passenger request (it may not be possible to meet all the requests). The input will be given in the following format nNcR n * n matrix indicating the distances to nearest locations. A sequence of N locations - indicating where the vehicle is at midnight. A sequence of R requests.

Algorithm :

- Requests sorted according to end time of the request
- After sorting it is checked that whether any cab is near the starting point of request(can reach from other vertex to that vertex before end time of the request)
- When the cab is full it is departed for dropping the passenger that is nearest to the current location of that cab.
- Procedure repeated for all requests
- At the end cabs which are not full but having some passengers will be departed for dropping the passengers

Data Structures used:

```
struct request{
	int reqId;
	int source;
	int destination;
	int startTime;
	int endTime;
	int flag;
	int cost;
	int pickupTime;
	int serviceCab;
	int dropTime;
};

struct cabs{
	int cabId;
	int location;
	int revenue;
	int passengers;
	int time;
	req *requests[100];

};
```

Variables used :

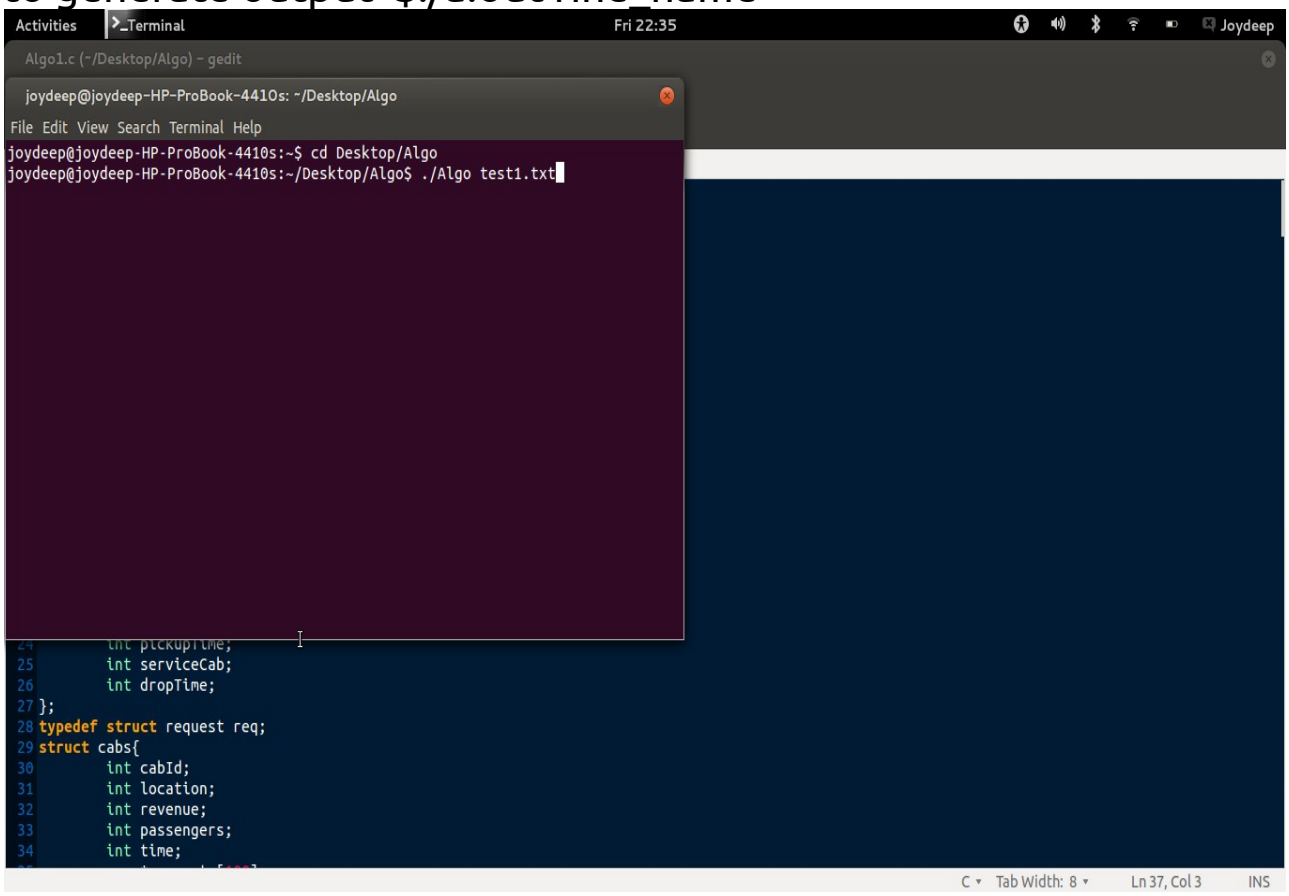
Structure: request	
Int reqId	Request id of each request
Int source	Source of the request
Int destination	Destination of request
Int flag	Ckeck whether the req was processed
Int startTimeI	When the req was made
Int endTime	When the req is to be seen
Int cost	Revenue generated by that request
Int pickupTime	When req is processed
Int dropTime	When req is dropped
Int serviceCab	Which cab processes it
Int nodes	Number of nodes in the graph
Int totalCabs	Number of Cabs
Int maxCap	Capacity of each cab
Int totalReq	Number of Requests
Structure Cabs	
Int cabId	Id of cab
Int location	Current location of cab
Int time	Stores the current time of each cab
Int revenue	Stores the Revenue of each cab
Int passengers	No of passengers present in the cab
Request *requests[100]	Stores the req a cab is serving.

Functions used:

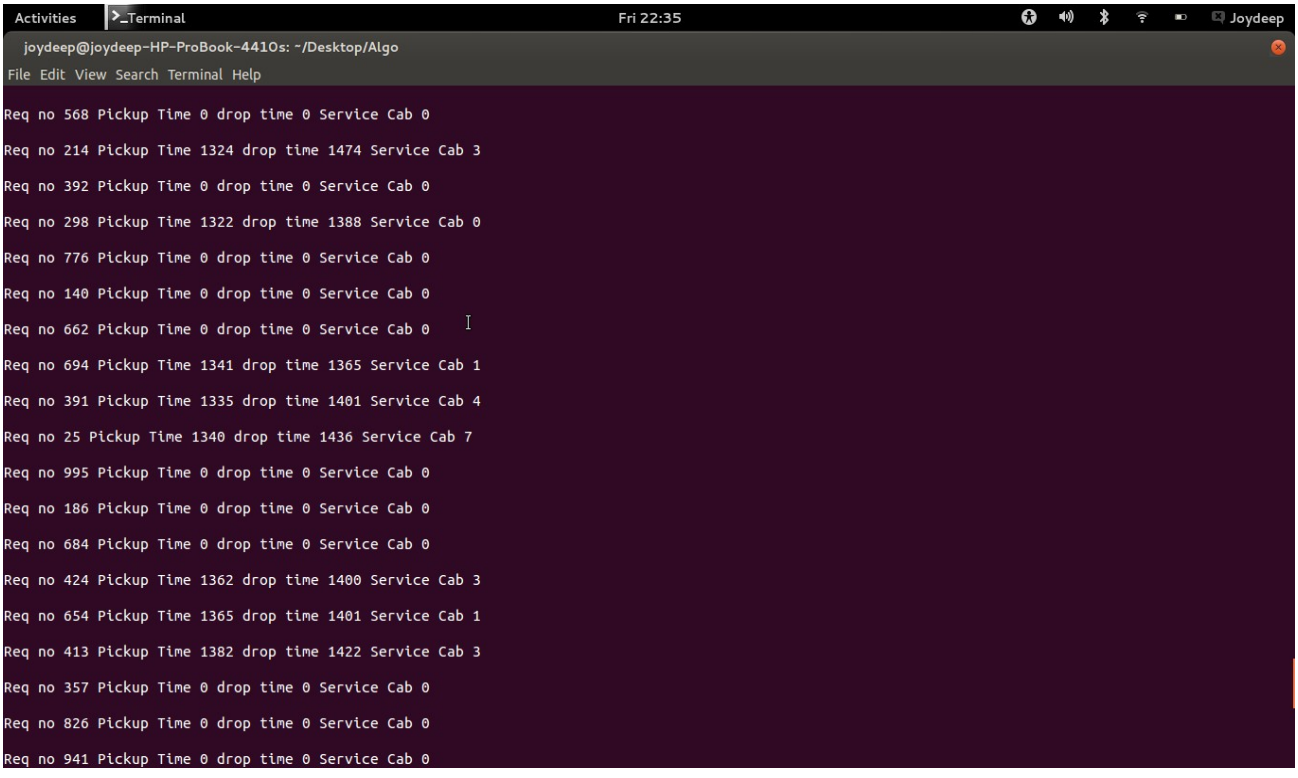
void CalculateShortestPath(int dist[100][100], int nodes)	Calculates all pair shortest path using Floyd warshals Algorithm
void SortList(req *requests[10000])	Sorts the req list based on end time
void DropPassengers(cab *cabss)	When the cab is full this function will drop a passenger which generates least revenue from that point
void FindCab(cab *cabss[1000], req *requests)	Finds the best possible cab for a given request
SelectReq(req *requests[10000], cab *cabss[1000])	Selects the best possible requests from the sorted req list
int main(int argc, char *argv[])	That calls all functions

Screen Shots

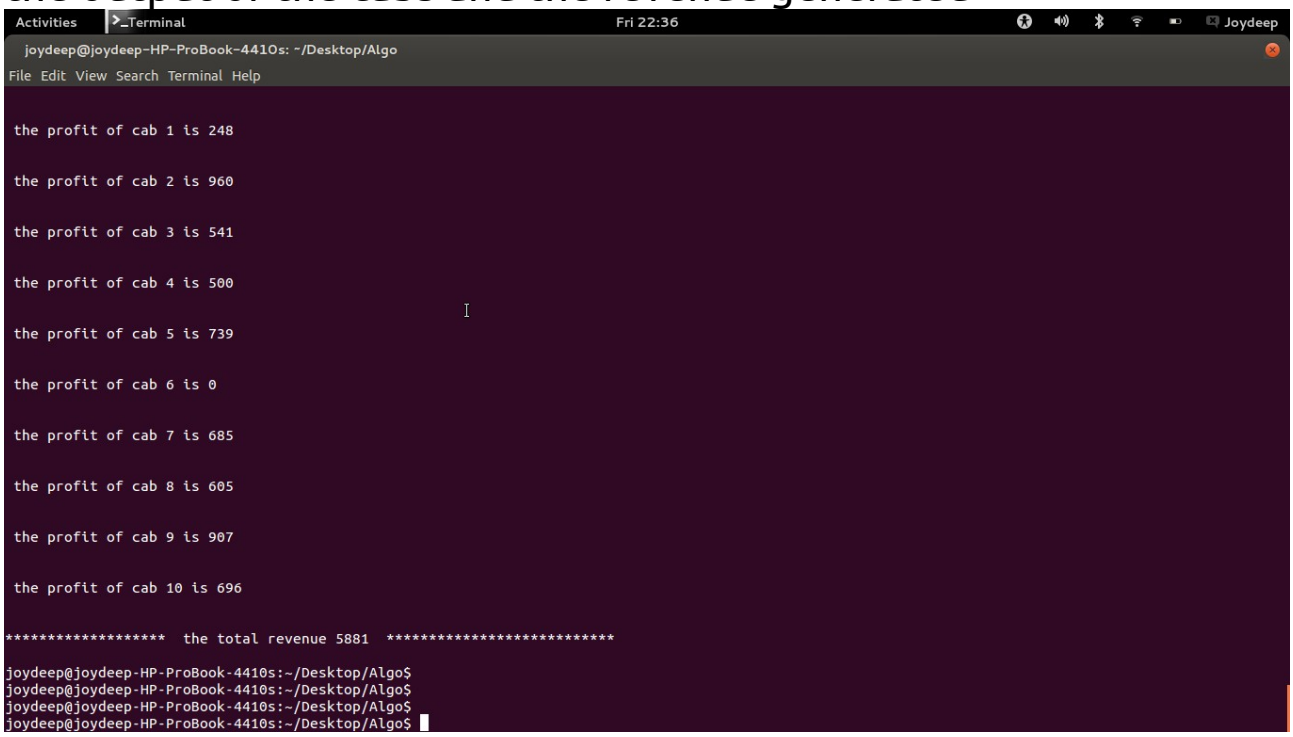
to generate output $./a.out fine_name



the output requests served



the output of the cabs and the revenue generated



Code Description:

Code consists of one file:

- MT2013062.c

The above file contains all the code

Running instructions:

./a.out <input-file-name>

Web Pages referred:

- http://www.wikipedia.org/
- http://stackoverflow.com/
- http://www.tutorialspoint.com