



Indian Institute of Information Technology
Kalyani, West Bengal

JAVA PROJECT ROAD TRAFFIC MANAGEMENT

A NUMBER OF DETAILS REGARDING THE PROJECT IS PRESENTED IN A
CLEAR AND CONCISE WAY

MADE BY

CHETAN MAHESHWARI

(ROLL NO. 39/CSE/17028/226

REG NO. 226)

PAWAN KUMAR

(ROLL NO. 39/CSE/17054/252

REG NO. 252)

4th SEMESTER

2nd YEAR

SUBMITTED TO:

DR. OISHILA BANDYOPADHYAY

2019

Contents

0.1	PROBLEM STATEMENT:	2
0.2	FUNCTIONALITY OF PROJECT:	2
0.2.1	Application Components used: -	2
0.3	APPROACH(ALGORITHM USED):	2
0.4	JAVA INBUILT PACKAGES USED:	3
0.5	DISCUSSION:	4
0.6	SCOPE OF IMPROVEMENT:	4
0.7	OUTPUT SCREENS:	4

0.1 PROBLEM STATEMENT:

GUI based expert system for road traffic management. The system should inform user about routing and re-routing of vehicles depending on congestion. (You can use any shortest path algorithm to implement this problem).

0.2 FUNCTIONALITY OF PROJECT:

Human drivers may perform replanning when facing traffic jams or when informed that there are expected delays on their planned routes. In this paper, we address the effects of drivers re-routing, an issue that has been ignored so far. We tackle re-routing scenarios, also considering traffic lights that are adaptive, in order to test whether such a form of co-adaptation may result in interferences or positive cumulative effects. An abstract route choice scenario is used which resembles many features of real world networks.

This project involves building an a desktop application on traffic management system. In the contemporary world, urban mobility is one of the unprecedented challenges to be tackled in the administration of a big city,there is a necessity for an effective system to combat with one of the unprecedented challenges.

0.2.1 Application Components used: -

Java, Net beans,Eclipse

0.3 APPROACH(ALGORITHM USED):

In this system the algorithm which is used to calculate the shortest route is Dijkstra's Algorithm using priority queue.

- It maintains a list of unvisited vertices.
- It chooses a vertex (the source) and assigns a maximum possible cost (i.e. infinity) to every other vertex. The cost of the source remains zero as it actually takes nothing to reach from the source vertex to itself
- In every subsequent step of the algorithm it tries to improve(minimize) the cost for each vertex. Here the cost can be distance, money or time taken to reach that vertex from the source vertex. The minimization of cost is a multi-step process.
- For each unvisited neighbor (vertex 2, vertex 3, vertex 4) of the current vertex (vertex 1) calculate the new cost from the vertex (vertex 1).

- For e.g. the new cost of vertex 2 is calculated as the minimum of the two ((existing cost of vertex 2) or (sum of cost of vertex 1 + the cost of edge from vertex 1 to vertex 2))
- When all the neighbors of the current node are considered, it marks the current node as visited and is removed from the unvisited list.
- Select a vertex from the list of unvisited nodes (which has the smallest cost) and repeat from 4th point.
- At the end there will be no possibilities to improve it further and then the algorithm ends
- The outer loop runs for $|V|$ times
- The inner loop runs for $|V-1|$ times for a complete graph as each vertex has $|V-1|$ edges
- Also, for each iteration of the inner loop we do an `extractMin` and a `reduceKey` operation for the vertex.
- Hence the total running time has an upper bound of $O(|V| * |V-1|)$.
- This is the upper bound, $O(|V|^2)$

0.4 JAVA INBUILT PACKAGES USED:

1. `javax.swing.*`
2. `java.awt.*`
3. `java.util.*`
4. `java.util.ArrayList`
5. `java.util.List`
6. `javax.imageio.ImageIO;`
7. `javax.swing.border.EmptyBorder;`
8. `java.awt.event.ActionEvent;`
9. `java.awt.event.ActionListener;`
10. `java.io.IOException;`
11. `java.awt.event.MouseEvent;`
12. `java.awt.event.MouseListener;`
13. `java.awt.event.MouseMotionListener;`

0.5 DISCUSSION:

It has been found that dijkstra's algorithm is much better than any other algorithm for determining the shortest path

- Results of our experiments show that re-routing indeed pays off from a global perspective as the overall load of the network is balanced.
- The user interface displays the traffic intensity in the particular road.
- We have (not implemented) the re-routing algorithm as the intensity of traffic is fixed by the user
- More the waiting time denotes more traffic in that region. (not yet implemented)
- Along with the traffic intensity index the end user is also provided with alternate routes using dijkstra's algorithm having lesser traffic intensity index. (not yet implemented)

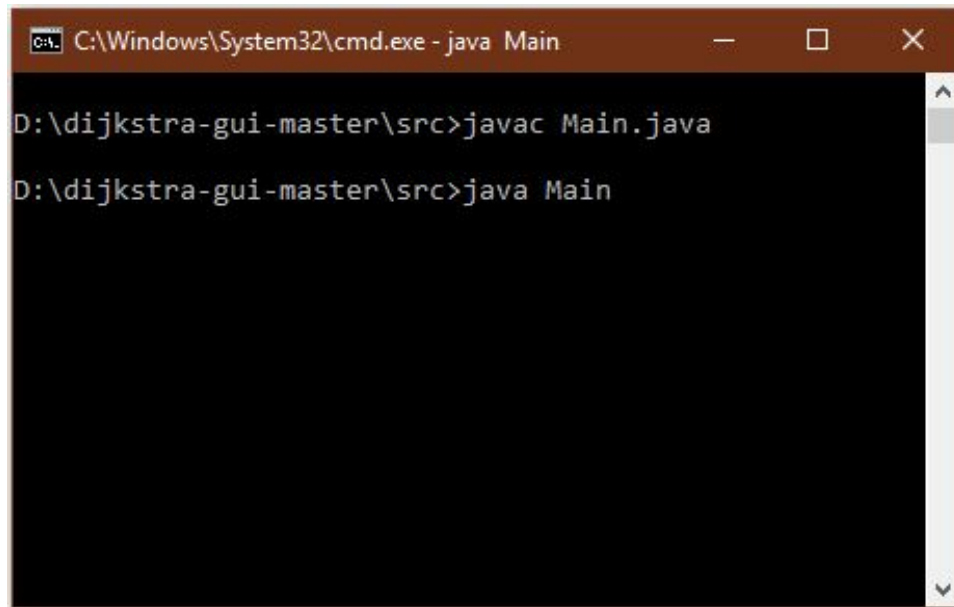
0.6 SCOPE OF IMPROVEMENT:

Based on the conclusion of the project research work, we recommend that we should implement the Dijkstra's Algorithm to find the shortest routes from their current place(source) to any other location of choice (destination) .

1. We are assuming here that there is no delay in the path due to red traffic light(that is there is green light at every place).
2. We can improve it by including the delay in path if there is red traffic light at some places.
3. We can include real time traffic intensity changing while travelling so then the route is to be re-routed.

0.7 OUTPUT SCREENS:

IS ON NEXT PAGE



```
C:\Windows\System32\cmd.exe - java Main

D:\dijkstra-gui-master\src>javac Main.java

D:\dijkstra-gui-master\src>java Main
```

Figure 1: Command Prompt commands to run Main Class.

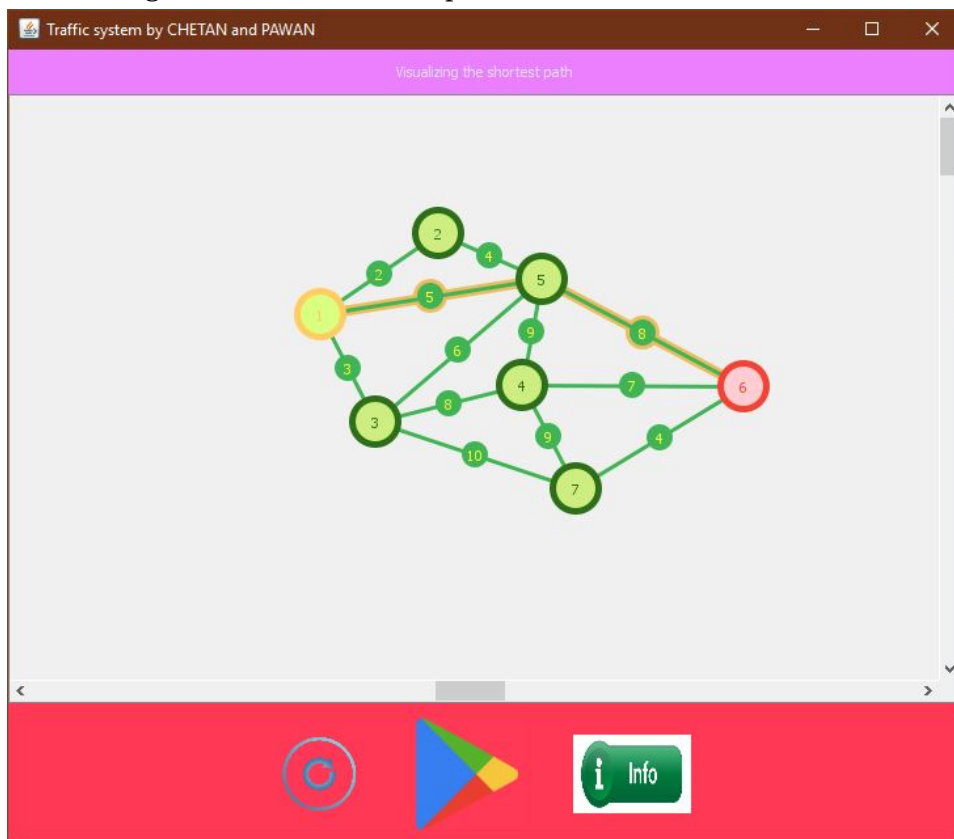


Figure 2: Swing program after running Main Class in Command Prompt.