Development of an application for navigation for low end mobile

A Project Report Submitted

In Partial Fulfillment of the Requirements

For the Degree of

Bachelor of Technology in Civil Engineering

By

Ankit Garg (Y7058)

To the

DEPARTMENT OF CIVIL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April, 2011

CERTIFICATE

It is certified that the work contained in the project report entitled, ``Development of an application for navigation for low end mobile'', by Ankit Garg (Y7058) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Name – Dr. Bharat Lohani
Department – Civil Engineering
I.I.T. Kanpur

April, 2010

ABSTRACT

This project is aimed to develop an application which will enable user to navigate using their low end mobile which lacks the facilities of GPRS and GPS. Due to limitations of low end mobile phones, text message seems a good mode of communication to such devices and application based on this communication have been developed.

Limitations of the application and scope for future improvement are also discussed towards the end.

ACKNOWLEDGEMENT

I am heartily thankful to my guide, Dr. Bharat Lohani, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the project.
Lastly, I offer my regards to all of those who supported me in any respect during the completion of the project.

Ankit Garg

INTRODUCTION

While venturing in an unknown zone it is sometime very difficult to get reliable direction to reach the destination. While there are many GPS (Global positioning service) based navigation system which can help regarding directions, it should be considered that the number of users using such mobile phones is very less in India when compared to the users using mobile phones which lack GPS facility.

In February 2011, there were 791.38 million mobile phone users. Out of which about 0.1% users have mobile phones which support GPS.

Thus a need for such application was felt which will allow users to get directions without using any other facility beside SMS (Short message service) which is available on every mobile phone.

STAGES INVOLVED

There are two trees involved in whole process. One is about handling and generating the database and other involves the handling of user input.

Two trees can be broken into different stages. Each stage will be discussed briefly in next section.

Below are the stages involved in generation handling the input provided by the user

1. User mentions the starting location and destination and transmits the message to server using SMS
2. GSM modem connected to the server receives SMS
3. SMS is retrieved from the GSM modem and sent for processing
4. SMS received is processed to identify the nodes near to the user location and destination
5. Once nodes are identified, Dijkstra's algorithm is used to identify the shortest path possible from start node to end road

6. Using the path directions are computed at each subsequent node encountered and landmarks on traversed route are identified
7. An SMS is compiled containing the information of turns to be taken and landmarks encountered and the SMS is transmitted back to the sender

Below are the stages involved in generation handling the input provided by the user

1. Images of the area of interest are taken from GoogleMaps
2. Using the ArcGIS, images are then converted to Shape files (.shp)
2. Shape files are converted to CSV files which are then read into PHP code
3. Using the data in CSV files, network of roads is defined using nodes

All the stages which might need an explanation are explained as follows:

User mentions the starting location and destination and transmits the message to server using SMS

A mobile application is designed which will ask the user about the landmarks present near the starting location and destination. Application is developed in J2ME (Java Micro Edition) and it is tested to support the *Symbian* operating system, which is most common in the low end mobile phones.

Javax classes have been used to design the user interface. Later communication ports were opened so that compiled string containing user input can be sent to the server.

The code developed is attached, named *sendSms.java*

GSM modem connected to the server receives SMS

I-300 Minicom GSM modem is used to receive the SMS from user. The modem is configured to run at *4800 baud rate.* The modem is connected to the laptop using the USB-serial port convertor. When received the SMS is stored in the SIM memory from where it is retrieved in later stage.

SSH command used to configure the GSM modem and communication port:

sudo stty -F /dev/ttyUSB0 4800

Here ***ttyUSB0*** should be replaced with the communication port active in the laptop/computer

SMS is retrieved from the GSM modem and sent for processing

When a new message is received a new event notification is sent to the communication port. Using the 'last modified time' for the communication port file, server is able to detect the notification.

SSH command:

echo "AT+CNMI=1,1,0,0,0" > /dev/ttyUSB0

This command is responsible for the notification received when new message is received.

Program responsible for retrieving the SMS is attached in file index.php

SMS received is processed to identify the nodes near to the user location and destination

Once the body of the SMS is received it is used to identify the starting node and destination nodes for the user. This process has many stages:

1. We first match the received landmark with the list of landmarks we have in the database. In case there is no exact match, most similar landmark is returned (similar in terms of spelling).
2. Road corresponding to the returned landmark is identified.
3. Distance of the landmark from both ends of road is calculated and end nearer to the landmark serves as starting/end node.
4. Dijkstra's algorithm is then used to compute the shortest route between the starting node and end node. (Explained in appendix A)

Using the path directions are computed at each subsequent node encountered and landmarks on traversed route are identified

First of all we oriented the user globally, i.e. we told the user in which direction he/she should be facing (East, West, North and South)

1. When going from one road to another, directions were calculated using the slope of the roads in Cartesian plane and sense of direction of road.

2. Right and left turns were provided when the slope was changing by more than 20 degree. If the change in slope was less than 20 degree user was advised to go straight.

3. All this is saved to the string, which is then transmitted to the user in form of SMS using the Minicom GSM modem.

Identification of area of interest and conversion to shape files

Area of interest is defined as the area in which the user wishes to navigate. In the development stage the area of interest was restricted to the *Acedemic Area* of the IIT Kanpur for the ease of development as the handling and development of database for large area of interest will take too much time and resources.

To develop the database, first of all we identified the area in Google maps. Using the Google maps we were able to identify the roads and landmarks, such as buildings, parks etc. Shape file was prepared containing the starting and end x,y coordinates of all roads. Each road was given a unique feature id. Length of each road was attached to the attribute table.

A separate shape file was created containing the landmarks identified in the area and feature ids of the roads associated with them.

Landmarks were categorized into two class on the basis of their recognition. Famous landmarks were put into class 1 while less famous landmarks were put into 2nd class.

Conversion from shape files to CSV format

Shape files were converted to CSV files using the *shp2xls* converter. The executable file of the software is attached with the report.

The conversion was done because limited support for the shape file reader class in PHP language. Reading a CSV file into PHP code was better supported and developed in PHP thus this conversion was implemented. CSV files were read into the PHP code using the *getcsv()* function with the appropriate parameters.

The relevant piece of code is attached as file named code1.php

Using the data in CSV files, network of roads is defined using nodes

From the road information provided in the CSV files a road network was generated showing the connectivity of each road with other road. Junction of any two roads is defined as a node. The first node was assigned to the road having feature id 1, from there onwards nodes were assigned to the junction according to order in which they appeared.

The relevant piece of code is attached as file named code1.php

Limitations and recommendations

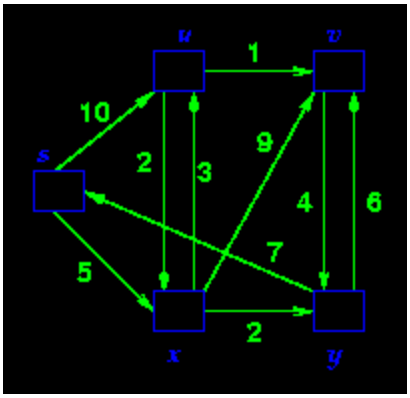
Due to the various problems faced during course of development, there are few limitations of this project. It was not yet able to use the actual shape files generated because those files contained very small roads for defining the curves and program was not able to handle them well while generating the direction string.
The process of sending/receiving of the SMS is not yet autonomous. Thus the project as a whole is not able to function but parts are working as intended.

I would like to recommend that PHP is not a good language for development of applications, thus this application should be reproduced in some faster language preferably C language.

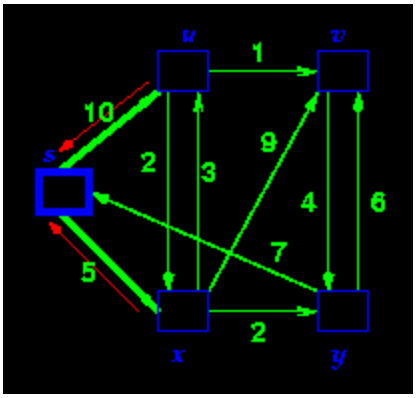
Some features could be added which can consider the mode of transportation to be employed by user while computing the directions. As route are sometimes different for motor vehicle and pedestrian.

APPENDIX: A

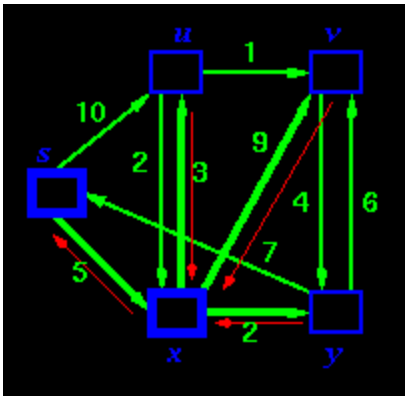
Step by Step operation of Dijkstra's algorithm



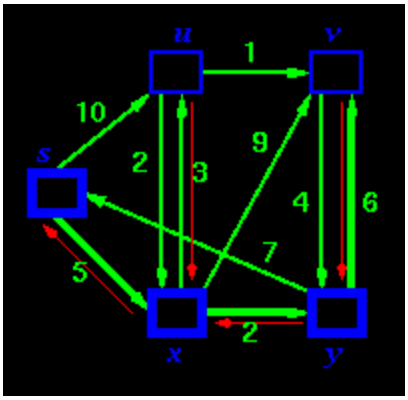
Step1. Given initial graph G= (V, E). All nodes have infinite cost except the source node, s, which has 0 cost.



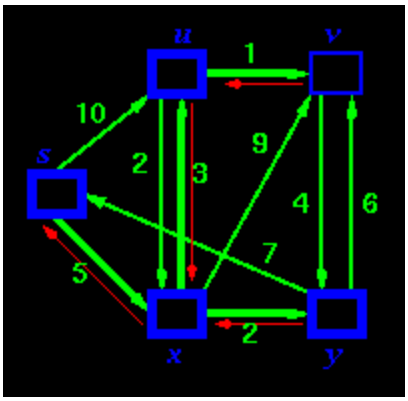
Step2. First we choose the node, which is closest to the source node, s. We initialize d[s] to 0. Add it to S. Relax all nodes adjacent to source, s. Update predecessor (see red arrow in diagram below) for all nodes updated.



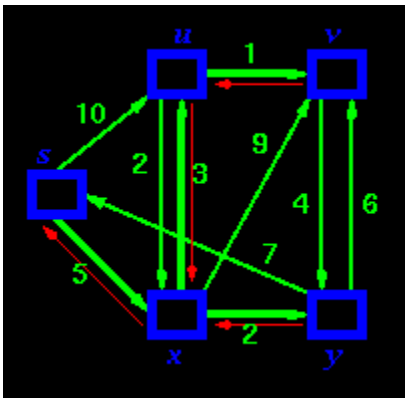
Step3. Choose the closest node, x. Relax all nodes adjacent to node x. Update predecessors for nodes u, v and y (again notice red arrows in diagram below).



Step4. Now, node y is the closest node, so add it to S. Relax node v and adjust its predecessor (red arrows remember!).



Step5. Now we have node u that is closest. Choose this node and adjust its neighbor node v.



Step6. Finally, add node v. The predecessor list now defines the shortest path from each node to the source node, s.