AUTOMATIC ROUTE FINDER FOR NEW VISITORS

By

MOHD SHIHAM BIN ADNAN

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

AUTOMATIC ROUTE FINDER FOR NEW VISITORS

by

Mohd Shiham Bin Adnan

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

Prof. Dr. P. A. Venkatachalam
Project Supervisor

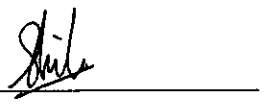
UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2006

Prof. Dr. P.A. Venkatachalam
Professor
Electrical & Electronic Engineering
Academic Block No. 22
Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh, Perak Darul Ridzuan, MALAYSIA.

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Mohd Shiham Bin Adnan

ABSTRACT

Shortest path finding algorithms are necessary in road network of a city or country for tourists and visitors. Automatic Route Finder tool for New Visitors is developed to help them to find or plan the shortest and effective way in order to reach their destination without loss of time. This project proposes a new visitor route model that is based on shortest path algorithms for road networks. Shortest path problems are among the best studied network flow optimization problems, with interesting applications in a range of fields. This study is based on Dijkstra's algorithm as the algorithm finds the most effective way to traverse an entire graph. The user is first need to select the starting location and the ending destination then the shortest path will be highlighted with the total distance between the two locations. If one of the roads in the first shortest path had problems, then the Automatic Route Finder will highlight the new shortest path from the two locations.

ACKNOWLEDGEMENTS

Thanks to almighty Allah S.W.T in giving me the opportunity to complete this final report submitted in partial fulfillment of the requirements for the Bachelor of Engineering (Hons) Electrical & Electronics Engineering. To my family members especially my father, my mother, my sister, and my brother, thank you for always being there and support me especially when I have to stay at UTP during the holidays.

I would like to take this opportunity to thank Electrical & Electronics Engineering Department of University of Technology PETRONAS (UTP) for providing me the opportunity to undergo such an inspiring project, and for awarding me this tremendously challenging yet interesting FYP title.

Special thanks go to Electrical & Electronics Department lecturers, especially supervisor Prof P. A. Venkatachalam, for his individually given guidance, motivation, advice and commitment to help the author during the completing the project. I would like to express my gratitude to PETRONAS, for the chances and sponsorship for the past 5 years in University of Technology PETRONAS.

I am indebted to the following people for the continuous support, encouragement, guidance and friendship through out working on this project. Last but not least, million thanks to all personal, whom directly or indirectly involved with the author upon completion of this project, thank you for the kindness and wonderful experiences.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

A map is a simplified depiction of a space, a navigational aid which highlights relations between objects within that space. Most usually a map is two-dimensional, geometrically accurate representation of a three-dimensional space. The road map that relates to this project is used to design a shortest route finder to help the new visitors to find the shortest and effective way in order to save the time to reach the destination.

As the first phase of this project, automatic route finder for new visitors has been designed to find several ways to different places of interest in Ipoh city. The city will indicate various destinations such as hospital, railway station, airport, government office, bus stations, etc, through the friendly medium of interactive multimedia. Information accuracy is very important in a system like this when various aspects of the city change rapidly. Shortest path algorithm is very important in this study as the project relates to calculate the shortest distance between the locations in the city. There are many algorithms such as Dijkstra, Bellman-Ford, A-star, Floyd-Warshall, Johnson's algorithm, etc.

Before selecting any of the algorithms, there is a need to compare them and find the best algorithm that suits for the shortest route project as it will be used in order to compute the shortest distance between two different nodes. There are many different operations that

can be done on graphs. Methods such as Bellman-Ford algorithm and Floyd-Warshall algorithm find the most effective way to traverse an entire graph. However, if the distance between two given vertices need to be calculated, an alternate method would be required.

The main function of the network layer is routing packets from the source machine to the destination machine. In most subnets, packets will require multiple hops to make the journey. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The shortest path problem is one of the most commonly encountered network-optimization problems in the study of transportation, communication and flow networks. One classical example is finding a shortest route from one location to another in a typical transportation problem. In this instance, model a road map as a weighted, directed graph, where the vertices represent intersections, the edges represent road segments between the intersections (nodes), and edge weights represent road distances.

In different networks, edge weights can be interpreted as metrics other than distances. They can be used to represent time, cost, penalties, or any other quantity that accumulates linearly along the path and that the author seeks to minimize. The cost of traveling from one vertex to another can be predetermined or computed when required. The figure 1.1 shows a typical subnet structure, or the example of the locations structure that will be used in this project. The nodes are the individual locations on the network.

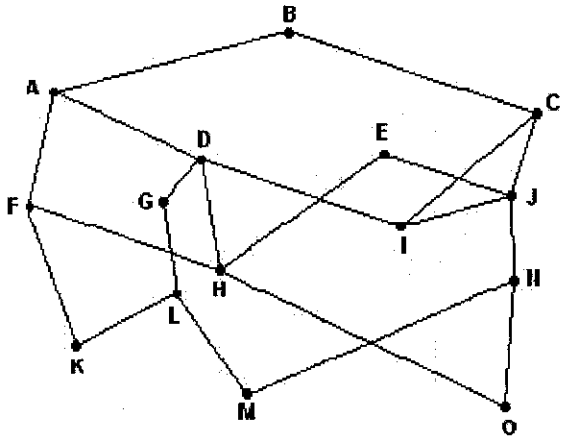


Figure 1.1: Typical Subnet Structure

The routing algorithm is that part of the network layer software responsible for deciding which line an incoming packet should be transmitted on. If the subnet uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If however virtual circuits are being used, routing decisions are made only when new virtual circuits are set up. Regardless of the case, there are certain properties that are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness and optimality.

Correctness and simplicity mean that the algorithm should be conceptually simple and error-free. Robustness implies that the algorithm should work irrespective of any change in the topology of the network or the failure of any network device. Stability means that the algorithm should not turn unstable no matter how complicated the routing equation. Fairness and optimality means that the algorithm should be non-discriminatory towards traffic of any particular type(s) or originating from any particular source(s), while at the same time taking efficient routing decisions that extract the best throughput from the network.

Routing algorithms can be grouped into two major classes: nonadaptive and adaptive. Nonadaptive algorithms do not base their routing decision on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from node I to node J (for all I and J) is computed in advance, off-line, and downloaded to the routers when the network is booted.

Adaptive algorithms, in contrast, change their routing decisions according to changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get their information from (i.e. from only neighbouring routers or all routers), when they change the routes (i.e. with change in traffic load or with change in topology), and what metric is used for optimization (e.g. distance, number of hops, transit time etc.).

1.2 Problem Statement

As mobile devices decrease in price and size, and increase in power, storage, connectivity and positioning capabilities, and tourists will increasingly use them as electronic personal tour guides. However they are not satisfied by being dumped with a lot of non-relevant information on their mobile devices.

For example, in Global Positioning System (GPS), the map that being used is very much detailed and complex but in this project, the system only used to show visually to the visitors the shortest distance from one selected starting location to the destination as from the author's point of view, the main reason why the user uses the shortest path routing system is to know the shortest path to save their time to reach the destination. The main context in this project is to help new visitors (whether they are local or outside tourist) who is not familiar of some places.

The most obvious applications arise in transportation or communications, such as finding the best route to drive between various destinations within Ipoh city or figuring how to direct packets to a destination across a network. The lack of interactivity presents the most dramatic drawback for almost all other forms of route finder. Signage that gives the chance for tourist to experience the movement of the map location to the desired destination might leave a lasting impression, can be thought provoking, and impacts moody behavior.

Shortest path problems are among the most studied network flow optimization problems, with interesting applications in a range of fields. This system need to quickly solve the shortest path problems but are typically embedded in devices with limited memory and external storage. Conventional techniques for solving shortest paths within large networks cannot be used as they are too slow or require huge amount of storage. Due to nature of routing applications, the project need flexible and efficient shortest path procedures, both from a processing time point of view and also in terms of the memory requirements.

The routing algorithm is a very simple and efficient that is widely used in many forms. The idea is to build a "graph" of the subnet, with each node of the graph representing a router and each arc of the graph representing a physical communication link between two nodes [1]. To choose a route between a given pair of routers, the algorithm simply finds the "shortest path" between them on the graph network. The figure 1.2 below shows the modified graph of the subnet, displaying the shortest routes from node B to all other nodes. Such a graph representing the shortest paths from all sources to a given destination is called a sink tree.

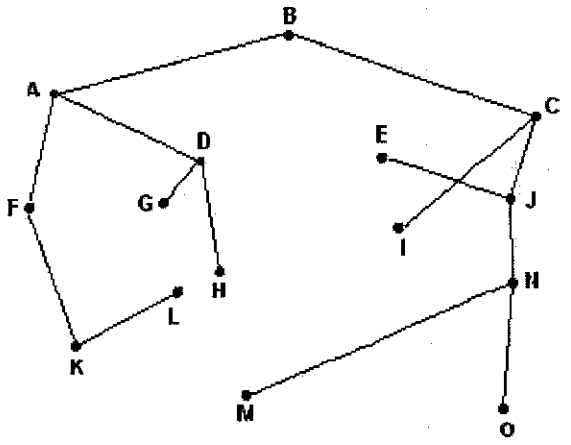


Figure 1.2: Modified graph of the subnet computed by node B

Remember that the sink tree is not computed just once, but is updated from time to time. This is because the individual nodes on the subnet may get changed from time to time, due to maintenance or improvement work and the algorithm must account for these changes. Also, excessive traffic on one particular link may render it unfavorable due to transmission delays, and hence an alternate link may be given preference.

The concept of a shortest path differs from algorithm to algorithm, depending on what system of measurement (called a metric) is used. One way of measuring path length is to measure the number of hops. Using this metric, the paths BAD and BAF are equally long. Another metric is the geographical distance in kilometers, in which case BAD is clearly longer than BAF. Many such other metrics are possible, such as mean queuing and transmission delay on a link, etc. In this project as the road network is used to calculate the shortest distance between several locations in a city (Ipoh) or a country (Malaysia), the metric used is the geographical distance whether in kilometers or meters.

1.3 Objectives

Most of the new visitors who need the route finder include of youngsters, family groups, senior citizen, disabled people, and also outside tourists. Several factors need to take into consideration in defining the objectives of this project such the level of understanding and level of interest.

The objectives of this project are:

1. To develop a road network in Malaysia with relevant and important bench marks to find the shortest route by using Dijkstra's shortest path routing algorithm as the algorithm find the most effective way to traverse an entire graph.

2. To provide the simplest, observable, and quick route finder which could help them to interact in order to know the shortest path from any starting location and ending destination.

3. To give the visitors with the user friendly design of the shortest route finder to ease the user to select the starting location and destination on the network.

4. To visually display the shortest path in the network on a lap-top or on a portable intelligent monitor fitted on the dash board of a vehicle.

1.4 Scope of Study

In this project, there are several scopes that define as the basis for this project. Consider the problem of the network analysis segmentation, that is, there are several nodes in a network where the nodes act as some places between locations in Ipoh city. In figure 1.3, let us assume that a new traveller has to go from a starting location 'S' to the destination, 'D'.

In network analysis method, there are many paths to reach the node 'D'. The user will select the places S and D. The system will display the road map of the area. Then it will highlight the available ways from any starting location 'S' to any destination 'D'. The system automatically will highlight the shortest path between the locations, the total distance and some benchmarks on the path.

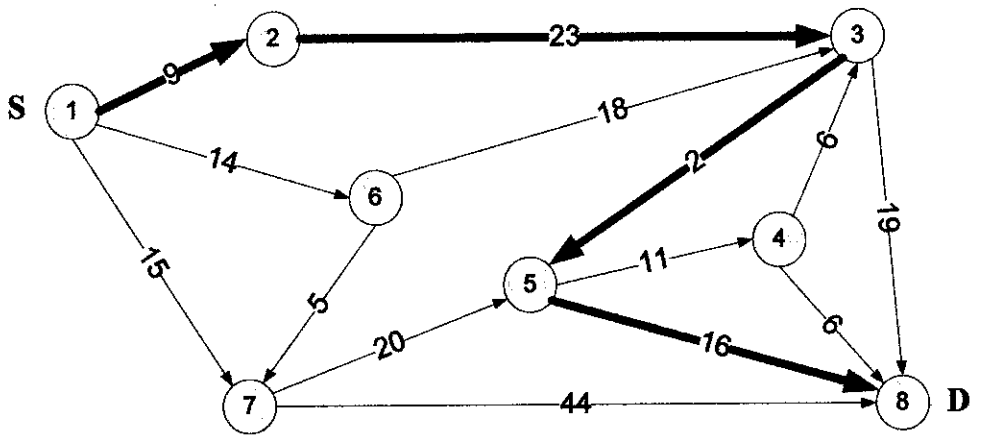


Figure 1.3: Highlighted Shortest Path

New visitors will need shortest route finder in order to find the shortest path from their starting location to the destination. By using the shortest path, it will minimize the time to reach the places. Thus, to solve these problems, this research adopts the Dijkstra's algorithm that is applied for maximizing some utilities under certain constraints. This algorithm progressively selects shortest distance between the selected locations.

1.4.1 The Advantages of the system

The system developed has several advantages. With full of information on 2D interactive road network, it could help the visitors to easily understand and use the system. The system will quickly highlight instantly the shortest path when the starting location and the destination are selected and at the same time indicate the distance.

The route finder may be a display monitor on the dash board similar to that of a speedometer.

The system that provides the user friendly application could increase the level of interactivity between the users. With this functionality, it will able to calculate a shortest path to provide the new visitors with better information regarding the places they desired to go in shortest time.

1.4.2 The Limitations of The System

The focus of this project is to have the simplest, intuitive, observable, and predictable route finder system that could help and guide all level of new visitors to navigate. But there are several limitations or drawback which the system could not encounter. The limitations of the system are:

1. The design of the system is just for a simple road network. But it can be improved to a detailed and complicated structure of the road. The road network allows the user to view the path between the locations highlighted. The system can also be made to zoom in or zoom out the road map as they desired on further research.

2. The system was planned to be applied with the portable processor based application but it will first need a prerequisite of the Microsoft Studio.Net as the program has been designed using the C.net programming language.

3. The system has to be modified when changes are made on the existing layout of roads. The system may need updating on the changes of the latest information of the city.

1.4.3 Feasibility of the Project within the Scope and Time Frame

In the first semester, the time used is about six months. The literature survey took two months and the remaining four months were used in developing the software system. In the second semester, the author has expanded the project for general implementation such as designing the shortest route finder for a city like Ipoh and for a bigger road network for a country like Malaysia. A good project management technique using the time constraint efficiently is adopted in order to complete the project successfully [*Appendix 1 and 2*].

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 Graph Theory

In Graph Theory, a graph is a collection of dots that may or may not be connected to each other by lines. A graph or network consists of vertices (locations) and arcs (roads) between the nodes. Such arcs may be directed or undirected [*Appendix 2*]. Graph theory is used in dealing with problems which have a natural graph/network structure, for examples:

- Road networks – nodes = towns/selected places/road junctions, arcs = roads
- Communication networks – telephone systems
- Foreign exchange/multinational tax planning a network of fiscal flows.

The minimum cost network flow problem is an example of a problem with a graph/network structure. Graph theory has a relatively long history in classical mathematics. In 1736 Euler solved problem of whether, given the map of the city of Konigsberg in Germany, someone could make a complete tour, crossing over all seven bridges over the river Pregel, and return to their starting point without crossing any bridge more than once [*Appendix 3*]. Euler's method formed the basis of what is known as graph theory, and which in turn paved the way for path finding algorithms [2].

Traditionally, network analysis, path finding and route planning have been the domain of graph theory where most algorithms find the in applications. However, it is not difficult to adapt these algorithms to the environment. Raster applications are more likely to be based on movement across a surface than movement along a network, since the general idea of finding the least cost path is linked to movement from cell to cell, and not along a finite line. Many researches have thus sought to improve the shortcomings of the raster approach and have developed various solutions and proposals. In order to appreciate their efforts, first, a synopsis of the conventions that encircle path finding algorithms is necessary.

2.2 Graph Theory and Fastest Paths

A network model can be defined as a line graph which is composed of links representing linear channels of flow and nodes representing their connections [3]. In other words, a network takes the form of edges (or arcs) connecting pairs of nodes (or vertices). Nodes can be junctions and edges can be segments of a road or a pipeline. For a network to function as a real-world model, an edge will have to be associated with a direction and with a measure of impedance, determining the resistance or travel cost along the network.

Traditionally, a Graphical Information System (GIS) represents the real world in either one of two spatial models, vector-based, i.e. points, lines and polygons, or raster-based, i.e. cells of a continuous grid surface. In a more generic sense, GIS is a "smart map" tool that allow users to create interactive queries (user created searchers), analyze the spatial information, and edit data. Geographic information systems technology can be used for scientific investigations, resource management, asset management, development planning, cartography, and route planning.

For example, a GIS might allow emergency planners to easily calculate emergency response times in the event of a natural disaster, or a GIS might be used to find wetlands that need protection from pollution. Since the modeling of network structures intuitively refers to lines and points, vector GIS has dominated the realm of network analysis. This research will show that network analysis is equally feasible in raster GIS.

2.3 Fastest Paths Algorithms

Because path finding is applicable to many kinds of networks, such as roads, utilities, water, electricity, telecommunications, computers, and alike, the total number of algorithms that have been developed over the years is immense.

In 1979, Pang and Deo set up a nomenclature of shortest path algorithms, describing as many as 222 different algorithms, dating back as far as 1958 [4]. Their paper provides an excellent classification scheme, a brief description of each algorithm and highlighted comparisons of particular algorithms.

Semantically one can distinguish between path finding in a fixed static network, with set costs for traversing the network, and path finding in a dynamic network, where the cost of traversing the network varies over the time of traversing.

One way of dealing with dynamic networks is splitting continuous time into discrete time intervals with fixed travel costs [3]. Thus, understanding shortest path algorithms in static networks becomes fundamental to working with dynamic networks.

2.3.1 Static Paths

The majority of published research on shortest path algorithms has dealt with static networks that have fixed topology and fixed costs. Given the computational restraints in the capacity of past computer systems this is not surprising. Not more than a decade ago, (Van Eck 1990) reports several hours as an average time for a computer to churn through an all-to-all calculation on a 250-nodes small-scale static network, and several days on a 16.000-nodes large-scale network.

The Dijkstra's algorithm explores all directions for economical paths from the starting node, thereby searching an unnecessary large search area. This led to the development of heuristic searches, among them the A* algorithm by Mitchell and Kiersey 1984 searches in the direction of the destination node. A* (pronounced "A star") is a graph search algorithm that finds a path from a given initial node to a given goal node (or one passing a given goal test).

It employs a "heuristic estimate" that ranks each node by an estimate of the best route that goes through that node. It visits the nodes in order of this heuristic estimate. The A* algorithm is therefore an example of best-first search. The name A* came about because it was originally presented as the optimal version (denoted by the *) of another algorithm, which was dubbed algorithm A. This avoids considering directions with non- favorable results and reduces computation time.

Another improvement is seen in the bi-directional search, computing a path from both origin and destination, and ideally meeting at the middle, which then terminates the search. However, as noted by Dreyfus (1969), in certain cases, the

number of iterations required may actually exceed the Dijkstra algorithm, or even produce a false result.

The Dijkstra algorithm seems to be preferred method in most of the literatures. It is in fact noteworthy that the Dijkstra algorithm has prevailed to the present date, proving its universal validity.

2.3.2 Dynamic Paths

Current literature suggests that dynamic network fastest path problems can be reduced to static fastest path problems if continuously varying link travel times are expanded for a time interval or given an estimated value. This makes the A-star and Dijkstra's algorithms applicable in both static and dynamic networks. Adaptations of Dijkstra's algorithm are frequently found in raster GIS application, thus it seems natural not to investigate more deeply into various dynamic path algorithms, as these are in most cases applied in graph theory, and not related to the research undertaken here.

2.4 Spanning Trees

The diagram shown in Figure 2.1 has four wells in an offshore oilfield (nodes 1 to 4) and an on-shore terminal (node 5). The four wells in this field must be connected together via a pipeline network to the on-shore terminal [2]. The various pipelines that can be constructed are shown as links in the diagram below and the cost of each pipeline is given next to each link.

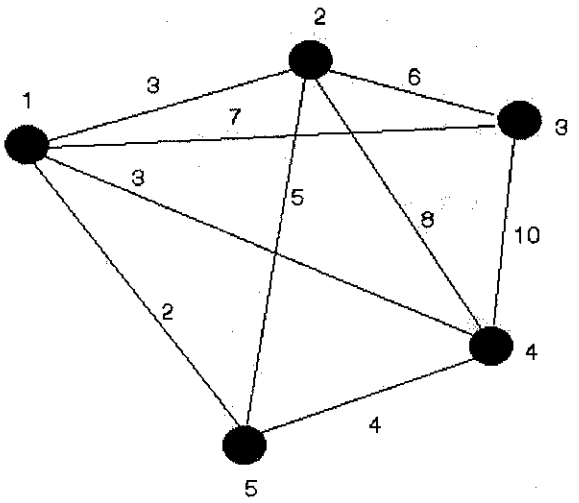


Figure 2.1: Spanning Trees [2]

It is clear that this particular problem is a specific example of a more general problem – namely given graph which links would we use so that the total cost of the links used is minimum and all the points are connected together. This problem is called the shortest spanning tree (SST) problem. Note that the phrase "minimal spanning tree" (MST) is also often used instead of the phrase "shortest spanning tree" (SST).

For example in Figure 2.1, one possible structure connecting all the points together would consist of the links 1-2, 2-3, 3-4, 4-5 and 5-1, but there are other structures where the total cost of the links used is smaller than in this structure

2.5 Shortest Paths in Raster GIS

In a raster GIS cartographic space is defined as a surface, where the value of a particular property varies over this surface. In order to adapt a network structure, each cell may be seen as a node linked to its eight neighbouring cells. The cell value of each node then can represent the cost of traversing this particular cell. This cost-of-passage surface is a grid where the values associated with the cells are used as weights to calculate least cost paths [5].

These weights may represent the resistance, friction or difficulty in crossing the cell and may be expressed in terms of cost, time distance or risk (Collischon and Pilar, 1999). Starting from a given destination cell, it is then possible to spread outward and calculate for each surrounding cell, the accumulated cost of travelling from any surrounding cell to the destination cell. From this accumulated surface it is then possible to delineate the shortest or least-cost path to the destination cell from any surrounding cell (Douglas, 1994), simply by following the path with the least accumulating friction.

This network adaptation from a surface has its shortcomings. A straight line indicating the shortest or least-cost distance from a starting node to a destination node must follow a zigzag line directed by the grid resolution, and thus only can approximate the correct distance as shown in Figure 2.2.

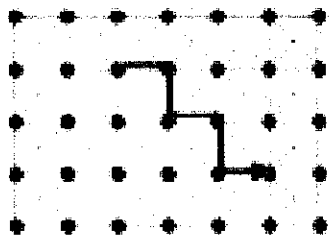


Figure 2.2: Tracing a path from cell to cell in raster GIS

2.6 Dijkstra's Shortest Path Algorithm [5]

Many more problems we might at first think be able to be cast as shortest path problems, making this algorithm a powerful and general tool. For example, Dijkstra's algorithm is a good way to implement a service like MapQuest that finds the shortest way to drive between two points on the map. It can also be used to solve problems like network routing, where the goal is to find the shortest path for data packets to take through a

switching network. It is also used in more general search algorithms for a variety of problems ranging from automated circuit layout to speech recognition.

Dijkstra's algorithm is an algorithm for finding a graph network, i.e., the shortest path between two graph vertices in a graph. It functions by constructing a shortest-path tree from the initial vertex to every other vertex in the graph.

The worst-case running time for the Dijkstra algorithm on a graph with n nodes and m edges is $O(n^2)$ because it allows for directed cycles. It even finds the shortest paths from a source node s to all other nodes in the graph. This is basically $O(n^2)$ for node selection and $O(m)$ for distance updates. While $O(n^2)$ is the best possible complexity for dense graphs, the complexity can be improved significantly for sparse graphs.

Dijkstra's algorithm works by visiting vertices in the graph starting with the object's starting point. It then repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined. It expands outwards from the starting point until it reaches the goal or the destination. Dijkstra's algorithm is guaranteed to find a shortest path from the starting point to the goal, as long as none of the edges have a negative cost. Dijkstra's algorithm is an example of a greedy algorithm, because it just chooses the closest frontier vertex at every step. A locally optimal, "greedy" step turns out to produce the global optimal solution. The algorithm works because it maintains the following two invariants:

For every completed vertex, the recorded distance (in visited) is the shortest-path distance to that vertex from starting node, v_0.

For every frontier vertex v, the recorded distance is the shortest-path distance to that vertex from v_0, considering just the paths that traverse only completed vertices and the vertex v itself. These paths called internal paths.

Dijkstra's algorithm works on the principle that the shortest possible path from the source has to come from one of the shortest paths already discovered. A way to think about this is the "explorer" model--starting from the source, we can send out explorers each traveling at a constant speed and crossing each edge in time proportional to the weight of the edge being traversed. Whenever an explorer reaches a vertex, it checks to see if it was the first visitor to that vertex: if so, it marks down the path it took to get to that vertex. This explorer must have taken the shortest path possible to reach the vertex. Then it sends out explorers along each edge connecting the vertex to its neighbors.

It is useful for each vertex of the graph to store a "prev" pointer that stores the vertices from which the "explorer" came from. This is the vertex that directly precedes the current vertex on the path from the source to the current vertex.

With slight modifications, Dijkstra's algorithm can be used as a reverse algorithm that maintains minimum spanning trees for the sink node. With further modifications, it can be extended to become bidirectional. The bottleneck in Dijkstra's algorithm is node selection.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This section describes details of method that will be used for the entire development of this project. The initial stage of the project involved extensive literature research on the topic of the project and its entire related subject. This section consists of two subtopics known as procedure identification, and tools or software used. The flow chart of system development procedure is shown in Figure 3.1.

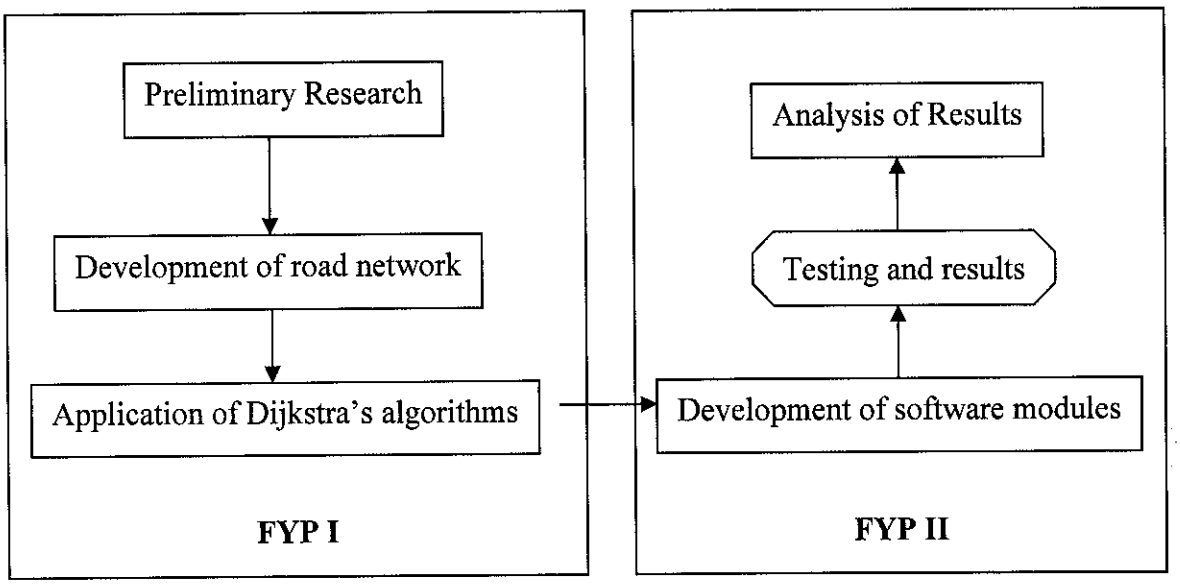


Figure 3.1: Flow chart of system development procedure

3.2 Procedure Identification

The design of the system is based on the implementation by Dijkstra's algorithm, named after its discoverer, Dutch scientist Edsger Dijkstra. It is an algorithm that solves the single source shortest path problem for a directed graph with nonnegative edge weights. Understanding the Dijkstra's algorithm to implement it through large network is most important area that needs a firm and comprehensive knowledge from the author. Therefore, the author is required to study all the important material regarding this subject that is related with the algorithm.

3.2.1 System Requirement and Scoping

As the objective of this project, it should meet the acceptance level by providing the visitors with shortest path routing system which is understandable and observable. The system also must show its functionality to provide visitors with enough knowledge on how to get into desired destination. By having the ability to provide enough knowledge to the visitors, it gives the system to gain advantage to achieve high level of understanding and depth of perceived for all levels of new visitors.

In order to achieve those objectives, shortest route finder should have ability of the system to search the required path quickly. This is one action of interactivity between visitors and the finder which give ability for them to interact which could increase level of attractiveness and also understanding. The project should meet the scopes which are simple and intuitive.

3.2.2 Task Analysis

Task analysis section describes on collection of service and action which it locates several activity in particular parts of the environment, define its artifact and also the tools involved for each service and action. For this system, it divides into several stages such as road network of the city, and the shortest path routing algorithm.

The first stage is to design a road network of the city as it will be the main environment to the user. The road map that being described here is the map of Ipoh city and the road network of Malaysia as the objectives of the project is to design an automatic route finder for new visitors in a city or a country for selected road network. The road network gives the user the ability to the visitors to view the buildings or benchmarks environment on the route when the vehicle moves.

This process occurred when the visitor selects the starting location and followed by the ending destination that he/she wants to go. The situation of the location will increase perceptions to the visitor to easy search where he/she is now located and which path that he/she can use as the road map will highlight the shortest path with the total distance and some benchmarks between the destinations. The following section will outline all the functions and algorithm that have been developed so far by the author in order to calculate the shortest path routing between the starting location and the destination.

3.3 Tools and Software

Knowledge of programming is essential in this project. Throughout this project, the algorithm that has been applied is processed by using Visual Studio.NET. Based on several stages which include of interactive application and virtual environment application, the tools need for each stage are differ but at the end of stages each of them will stitch to each others on one main application. The shortest route finder will be designed using windows applications with C# (C sharp) programming. The project will look and act like standard windows programme.

The first step before developing the algorithm is to design the map of Ipoh city as test case 1 and the map of Malaysia as test case 2. Before start drawing the map, several data and information that contain the distance between several main buildings have been collected. The nodes in Figure 3.2 indicate the main buildings in the Ipoh city while the paths between the nodes indicate the distance from each node in km. Figure 3.3 shows the nodes and the paths in Malaysian State Road Network.

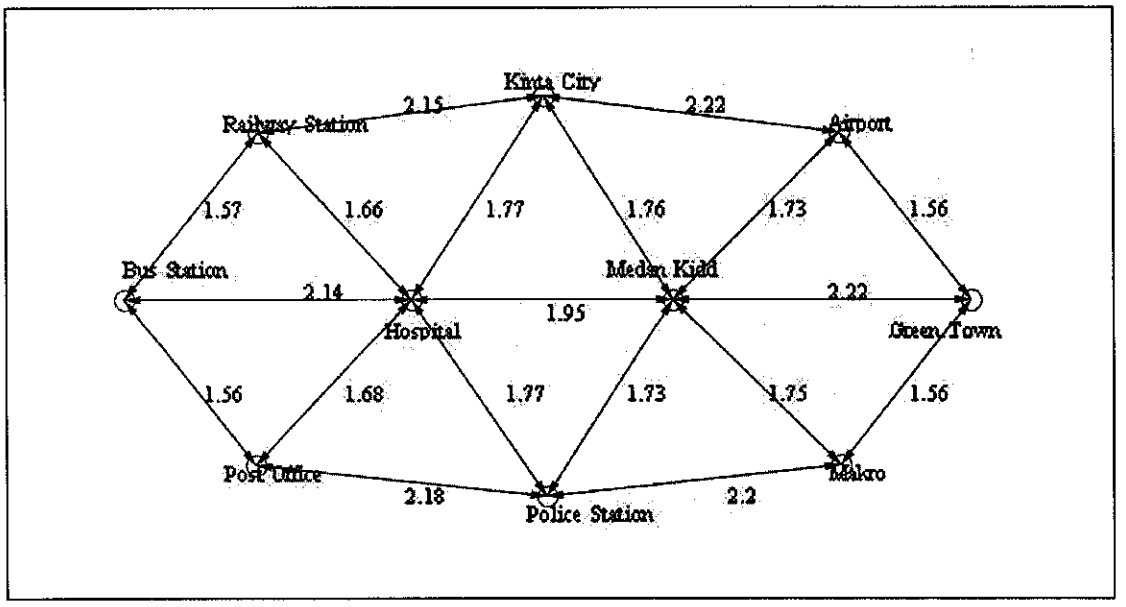


Figure 3.2: Ipoh City Road Network

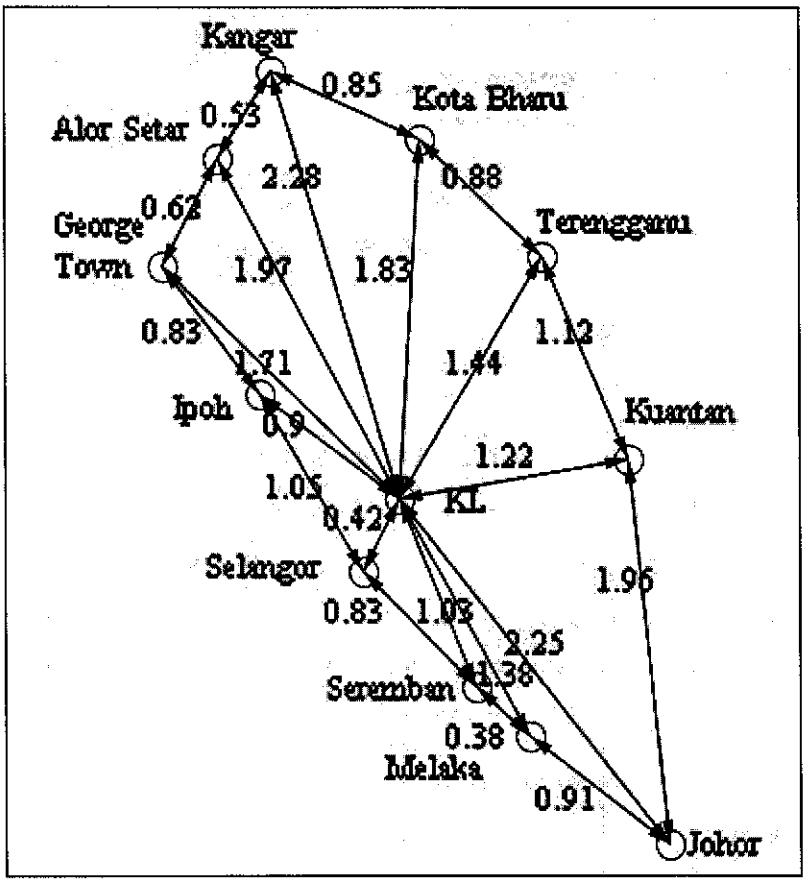


Figure 3.3: Malaysian State Road Network

The next step after the complete design of Ipoh city and Malaysia road network is to develop the shortest path routing algorithm. For the shortest path routing using Dijkstra's algorithm, the router builds a graph of the network and identifies starting location node as V1 and ending destination node as V2. Then it builds a matrix, called the adjacency matrix. In this matrix, a coordinate indicates weight. For example, [i,j] is the weight of a link between Vi and Vj, this weight is identified as infinity. The router builds a status record set for every node on the network. The record contains three fields:

- Predecessor field – The first field shows the previous node
- Length field – The second field shows the sum of the weights from the source to that node
- Label field – The last field shows the status of the node. Each node can have one status mode: "permanent" or "tentative"

The router initializes the parameters of the status record set (for all nodes) and set their length to "infinity" and their label to "tentative." The router sets a T-node. If V1 is to be the source T-node, the router changes V1's label to "permanent." When a label changes to "permanent," it never changes again. A T-node is an agent and nothing more. The router updates the status record set for all tentative nodes that are directly linked to the source T-node. The router looks at all tentative nodes and chooses the one whose weight to V1 is lowest. The node is then the destination T-node. If this node is not V2 (the intended destination), the router goes back to the status update. If this node is V2, the router extracts its previous node from the status record set and does this until it arrives at V1. This list of nodes shows the best route from V1 to V2. The flowchart in Figure 3.4 shows simplified implementation method of the Dijkstra's algorithm.

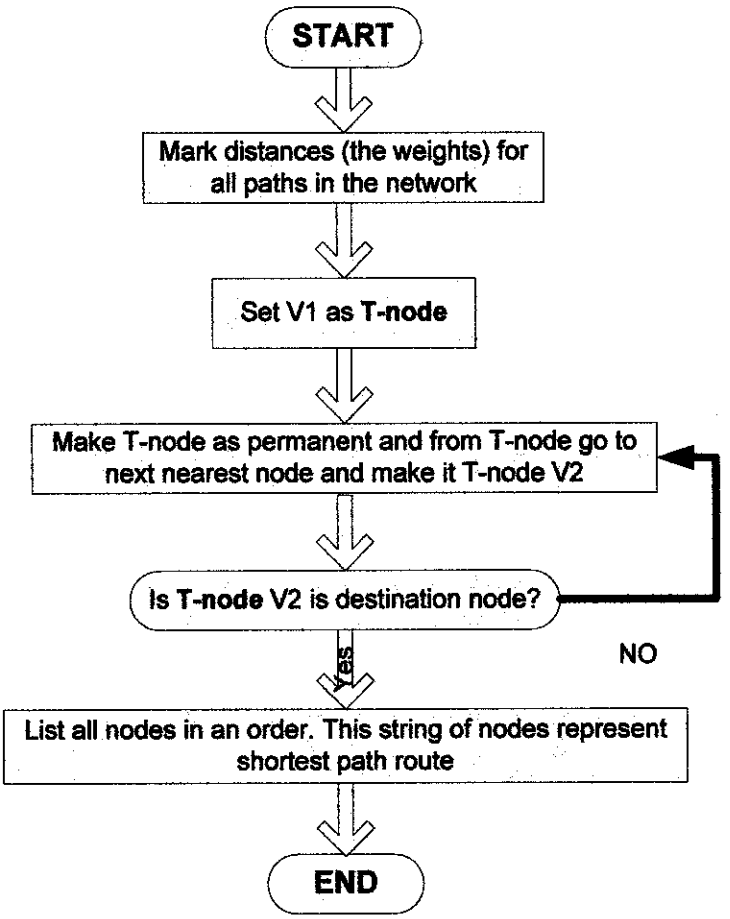


Figure 3.4: Flow chart of Dijkstra's Algorithm

In order to easily understand about the routing algorithm between locations in the road network, simple diagrams about the Dijkstra's have been understood [*Appendix 4*]. Here the system wants to find the best route between A and E. There are six possible routes between A and E (ABE, ACE, ABDE, ACDE, ABDCE, ACDBE), and it is obvious that ABDE is the best route because its distance is the lowest [6]. From the network below, the source node (A) has been chosen as T-node, and so its label is permanent (permanent nodes with field circles and T-nodes with the → symbol.

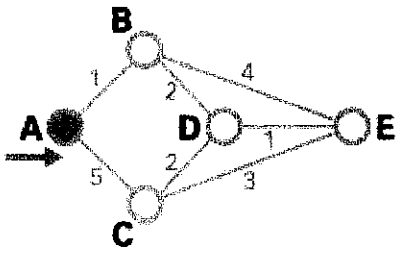


Figure 3.5: Node A as T-node

Next, the status record set of tentative nodes directly linked to T-node (B,C) has been changed. Also, since B has less weight, it has been chosen as T-node and its label has changed to permanent (see network below).

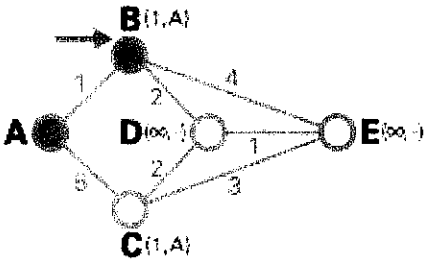


Figure 3.6: A to B

Then, the status record set of tentative nodes that have a direct link to T-node (D,E) has been changed. Since D has less weight, it has been chosen as T-node and its label has also changed to permanent as shown in network below.

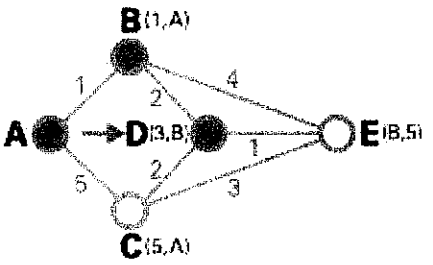


Figure 3.7: A to B to D

Lastly, it does not have any tentative nodes, so it just identifies the next T-node. Since E has the last weight, it has been chosen as T-node.

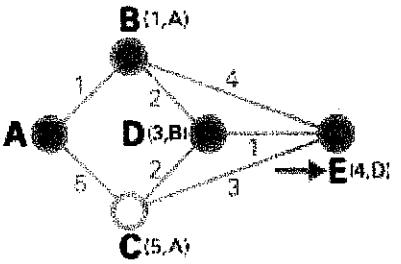


Figure 3.8: A to B to D to E

E is the destination, so it stops there. The previous node of E is D, and the previous node of D is B, and B's previous node is A. So the best route is ABDE. In this case, the total weight is **4** (1 + 2 + 1). If a router gives the wrong information to other routers, all routing decisions will be ineffective.

This algorithm makes an important addition to Dijkstra's original algorithm. Along with the minimum number of hops, it also uses a link length metric to compute the shortest path. As per Dijkstra's algorithm, it first tries to find a route with the minimum number of hops between the source and destination. However, it does not stop when the first valid route is found. It tries to find alternative routes with the same number of hops. If it finds such alternative routes, it then calculates the total physical length of all the routes separately and compares them. The shortest route is chosen. The program finally displays the shortest path in a convenient, easy-to-read way.

For the implementation of the algorithm, the author use a two-dimensional array of structures whose every element represents a link between any two nodes 'i' and 'j'. Each structure has two parameters - a "link presence" parameter and a "link length" parameter. The presence parameter indicates the presence or absence of a link between the two nodes 'i' and 'j'. If the value of link [i,j].presence is '0', it means there is no link between 'i' and 'j'. If it is '1', a link is present. The second parameter length indicates the physical length of the link, if present, in kms. If there is no link, its value is '0'.

Below is the declaration of the array and structure:

```
//Define a structure to store information about each link
struct linkdata
{
int presence;        //Tells  if a link is present or not
int length;          //Tells  if a link is present or not
};

//Define a 2-D array of type 'data' and size 'nxn'
//Parameter 'MAX' is the maximum permissible number of nodes in the subnet
linkdata subnet[MAX][MAX];
```

When writing the program, the author has to enter the necessary details of the subnet, namely the number of nodes, the various links present and their lengths. Below is the coding that will be used in updating the details:

```
  /Form array map of subnet
//Parameter 't' is the total number of nodes in our subnet (<=MAX)
cout<<"\n\nEnter the total number of nodes in the subnet: ";
cin>>t;
for(i=0;i<t-1;i++)
{
cout<<"\nEnter the number of nodes to which node "<<i+1<<" is connected: ";
cin>>x;
cout<<"\nEnter the numbers of those nodes one by one:";
for(j=0;j<x;j++)
{

cout<<"\n"<<j+1<<": ";
cin>>y;
subnet[i][y-1].presence=subnet[y-1][i].presence=1;
cout<<"\nEnter length of link from "<<i+1<<" to "<<y<<": ";
```

```
cin>>z;
subnet[i][y-1].length=subnet[y-1][i].length=z;
}
}
```

Designing the user interface will be the next step after the development of the shortest path algorithm. A sketch in the screen that the user will see when running the shortest route finder is needed before planning the code. On the sketch, it shows the forms and all the controls that will be used in the application. It also indicates the names of the form and each of the objects on the form. The graphical interface provides a visual representation of the subnet map. The individual node refers to the different locations of the map. It then connects all those nodes that have links between them with lines. Finally the shortest path is the line that been highlighted. Figure 3.5 is the final design of the user interface of Ipoh City Map.

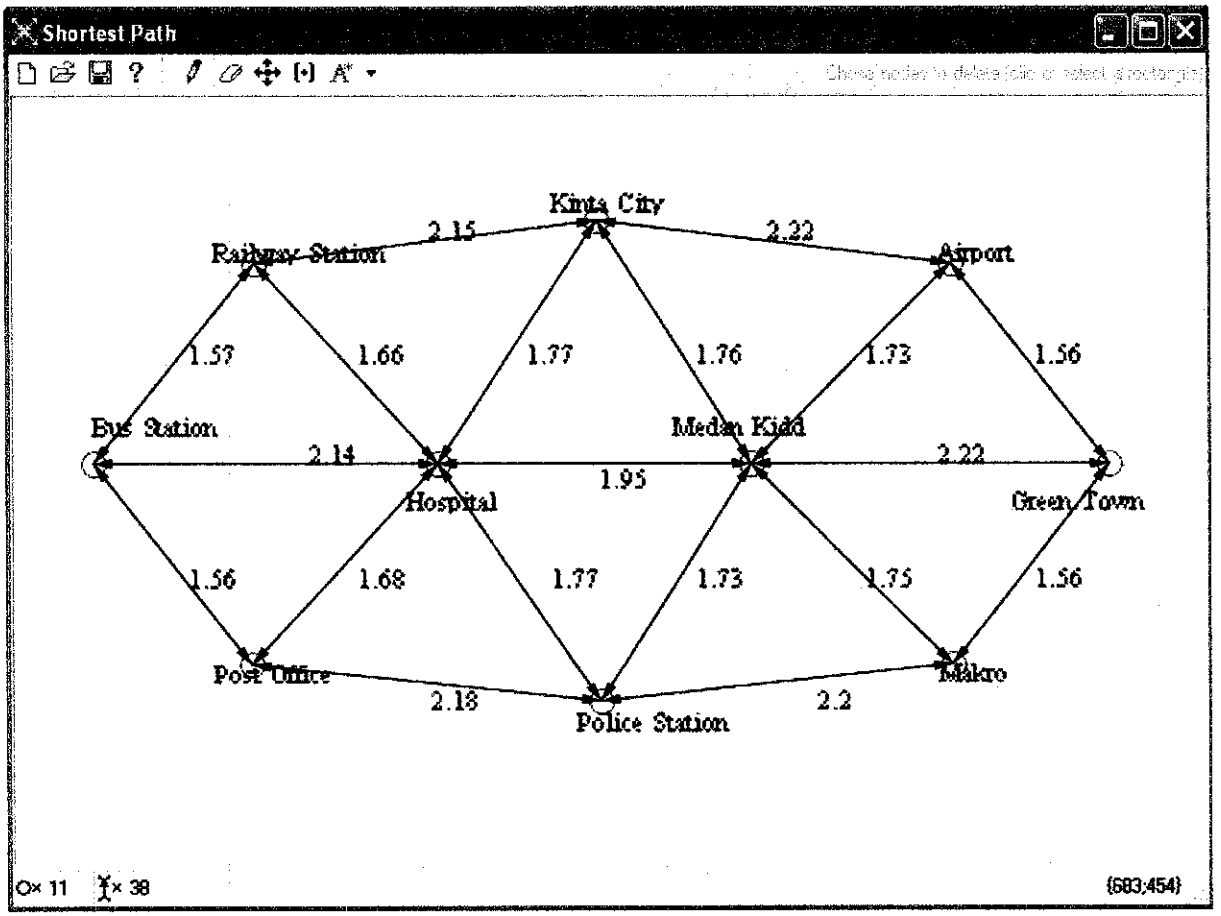


Figure 3.9: User Interface Design (Ipoh City Map)

After designing the user interface, the process can proceed with the planning of C# code. The classes and methods that will execute when the projects run need to be planned in this step. During the planning stage of the C# code, the author will write out the actions using pseudocode, which is an English expression or comment that describes the action. In the project, the author plans for the event that occurs when the user clicks on the Exit button. The pseudocode for the event will be terminating the project.

CHAPTER 4

RESULTS AND DISCUSSION

Automatic route finder was tested by using the road map of Ipoh city as test case 1and road map of Malaysia as test case 2 as shown in Figure 3.2 and Figure 3.3 respectively. This system requires the visitors to select the starting location and then followed by the ending destination as they desired. Once the visitors select the ending destination, the system will automatically highlight the shortest distance from the starting location and the destination. Then the route finder will display the total distance of the shortest path.

4.1 Test Case 1: Ipoh City Road Network

Once the traveller gets access to the system, they will see the main window of the shortest path routing. In this window it consists of two frames which are current path and also the boxes which will be the road map of the Ipoh city. By now, the system has been designed to find the shortest path from the starting location to the destination in Ipoh city. The light blue star in the road network presents the starting location while the dark blue star represents the ending destination. In test case 1, suppose the traveller needs to go from starting location (bus station) to destination (Green Town). There are several paths between bus station and Greentown but not all the paths are the shortest path and it is obvious that bus station – hospital – Medan Kidd – Green Town is the best route because its distance is the lowest as shown in Figure 4.1. The total distance follow the path from Bus Station – Hospital – Medan Kidd – Green Town is 2.14 + 1.95 + 2.22 = 6.31km as can be seen from the bottom right of the window.

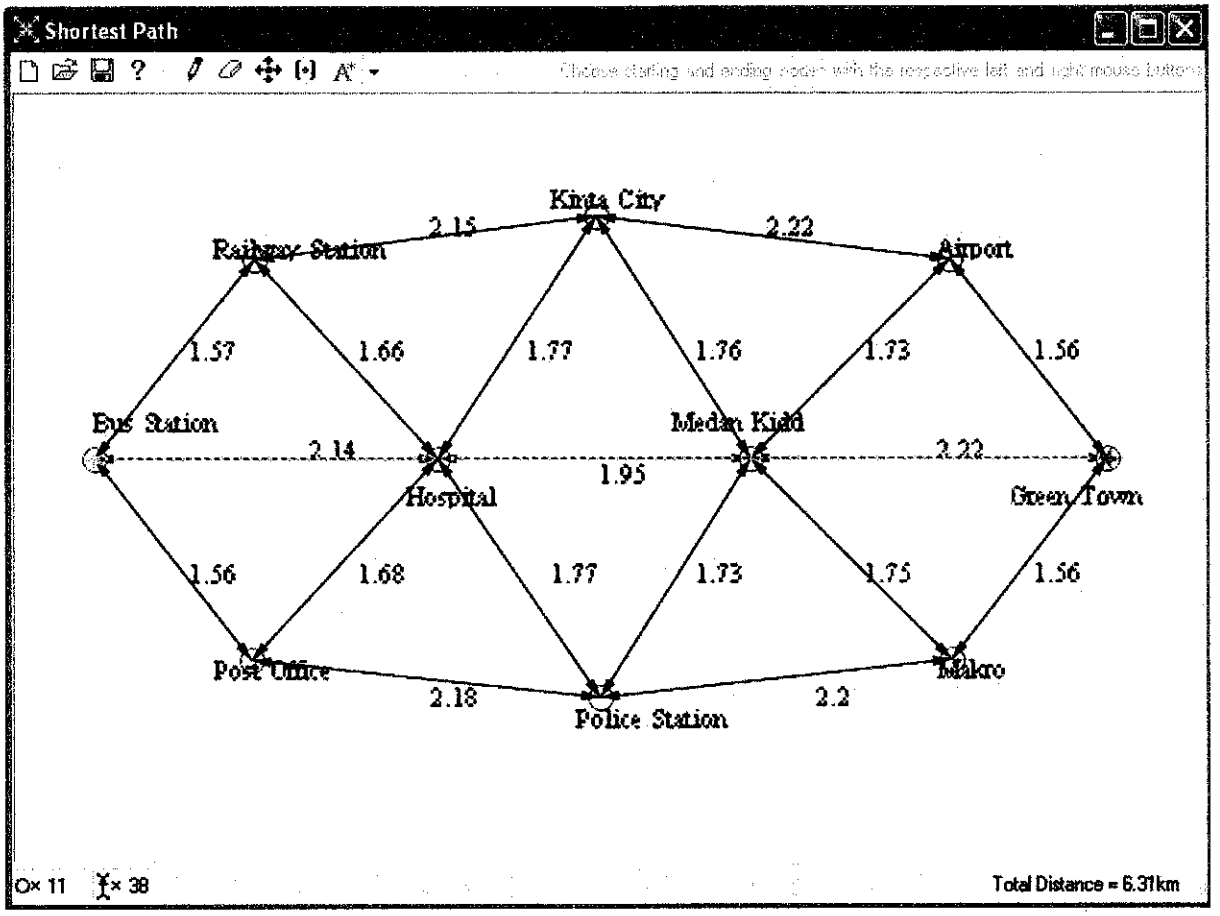


Figure 4.1: Test Case 1 (Road Network of Ipoh City)

As we know the road network of main city like Ipoh, Peneng, and Kuala Lumpur, their road network did not have only two ways of direction but sometimes they have only one way direction at certain roads. In this case it will explain how the shortest route finder finds the next shortest path if one of the roads only has one way of direction. From Figure 4.2 we can see that the road from Green Town to Medan Kidd has only one way of direction so that the route finder cannot go through Medan Kidd to Green Town but must use the other alternative roads which are using through the Airpot road or through the Makro. As a result as shown in Figure 4.2, the shortest route from Bus Station to Green Town is Bus Station – Hospital – Medan Kidd – Airport – Green Town and the total distance is 2.14 + 1.95 + 1.73 + 1.56 = 7.38km as shown at the bottom right of the window.

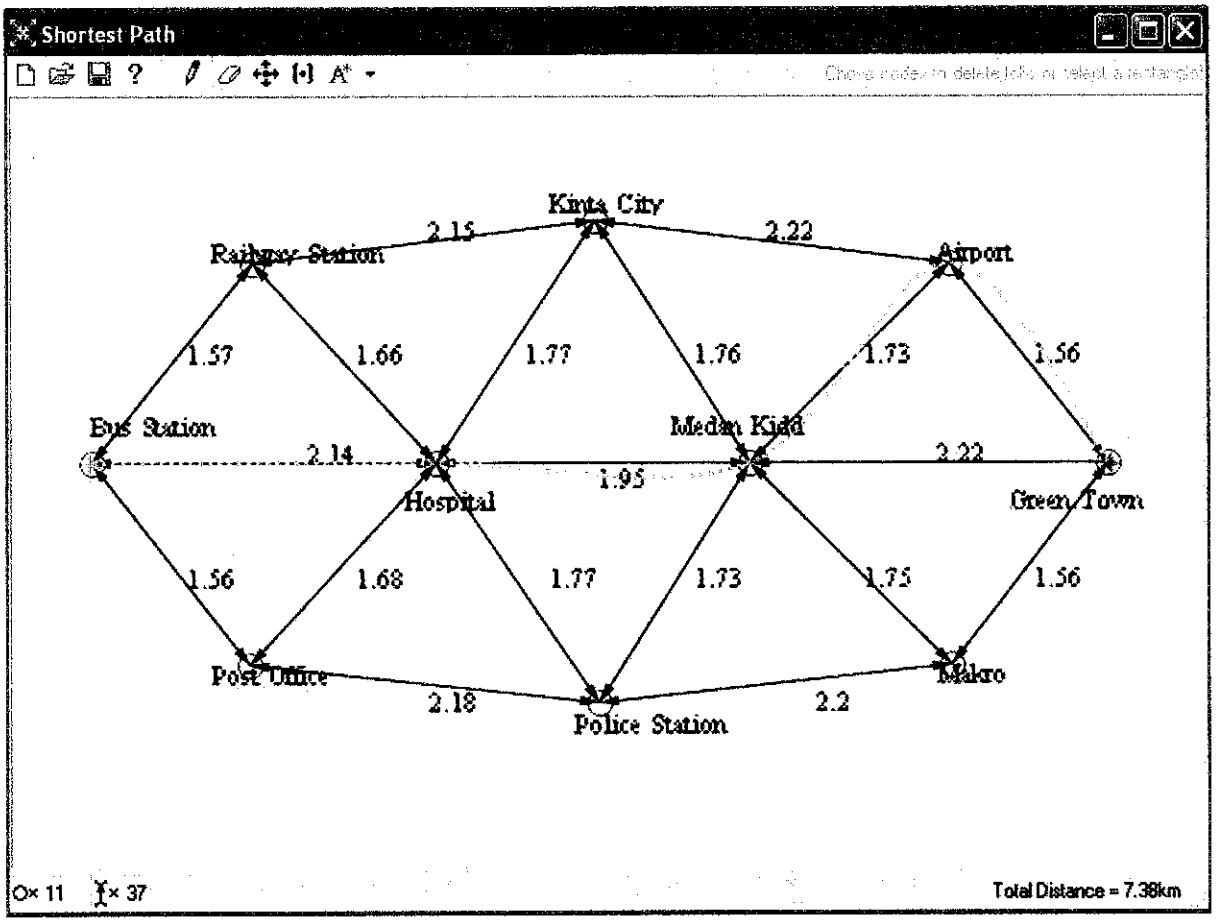


Figure 4.2: Road Network of Ipoh City (One way of direction)

Next in this test case, it will describe about how the shortest route finder will find the next shortest path between two different locations if one of the road in the first shortest path has a repair or a problem. In this case the road network of Ipoh city has been used. As we can see in Figure 4.1 the first shortest path between Bus Station and Green Town is Bus Station – Hospital – Medan Kidd – Green Town which is 6.36km. Let say if the road from Bus Station to Hospital has a repair work so that the road will be closed and we need the new shortest path. In this project the traveller need to delete the road from Bus Station to Hospital and the route finder will display or highlight the new shortest route from the locations. In Figure 4.3 the new shortest route is Bus Station – Railway Station - Hospital – Medan Kidd – Green Town and

the total distance is 1.57 + 1.66 + 1.95 + 2.22 = 7.4km as shown at the bottom right of the window.

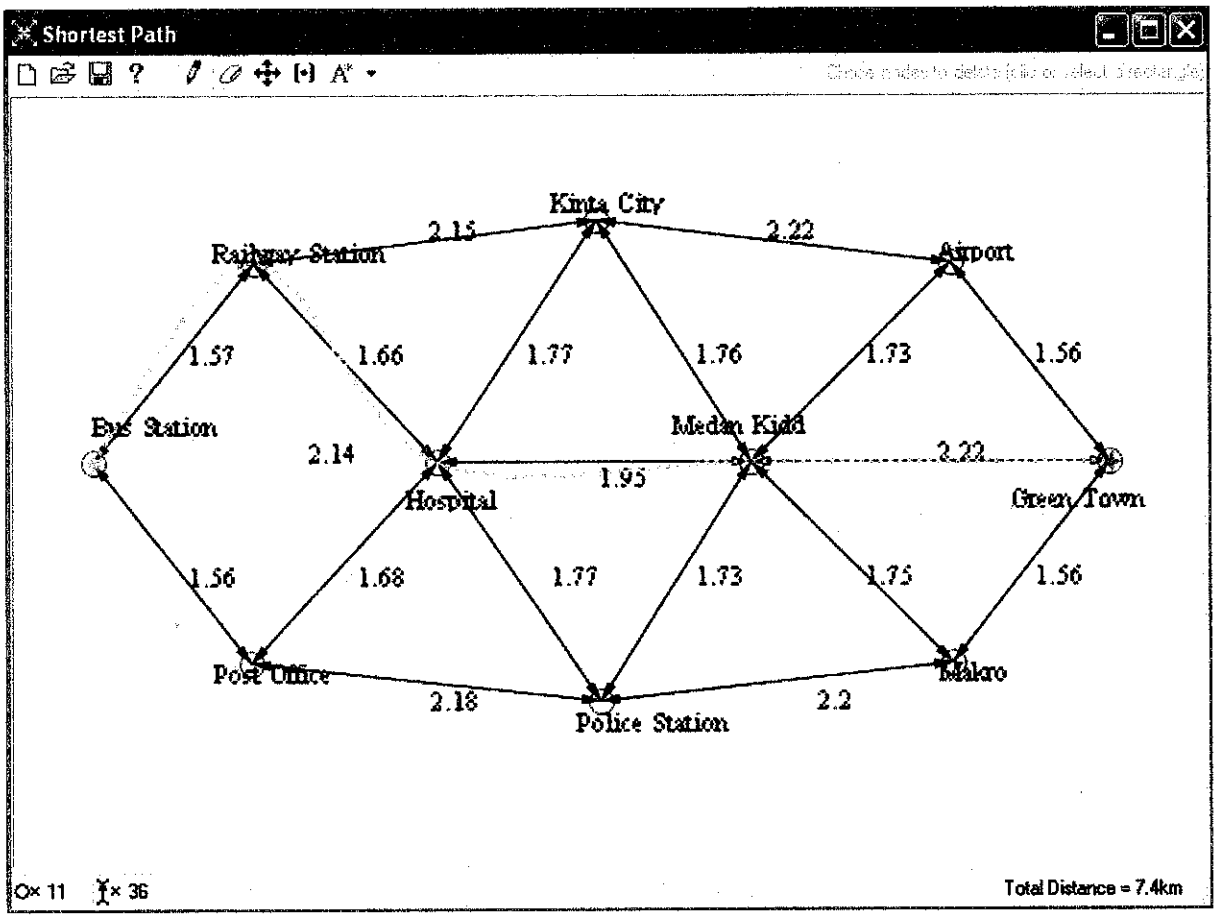


Figure 4.3: Road Network of Ipoh City (Road from Bus Station and Hospital on repair)

4.2 Test Case 2: Malaysian State Road Network

In test case 2, the state road network of Malaysia has been used. As we can see in Figure 4.4, the traveller starting location is at Ipoh city which is indicated with the light blue star on the network. The traveller needs to find the shortest path from starting location Ipoh city to the ending destination which is Johor indicated with the dark blue star. When the traveller selects the destination, the shortest path will be

highlighted and the system will display the distance between the two locations. Same as test case 1, there are also several paths between Ipoh and Johor and the result of shortest path is Ipoh – KL - Johor which the total distance is approximately 0.9 + 2.25 = 3.15km as shown at the bottom right corner in figure 4.4.

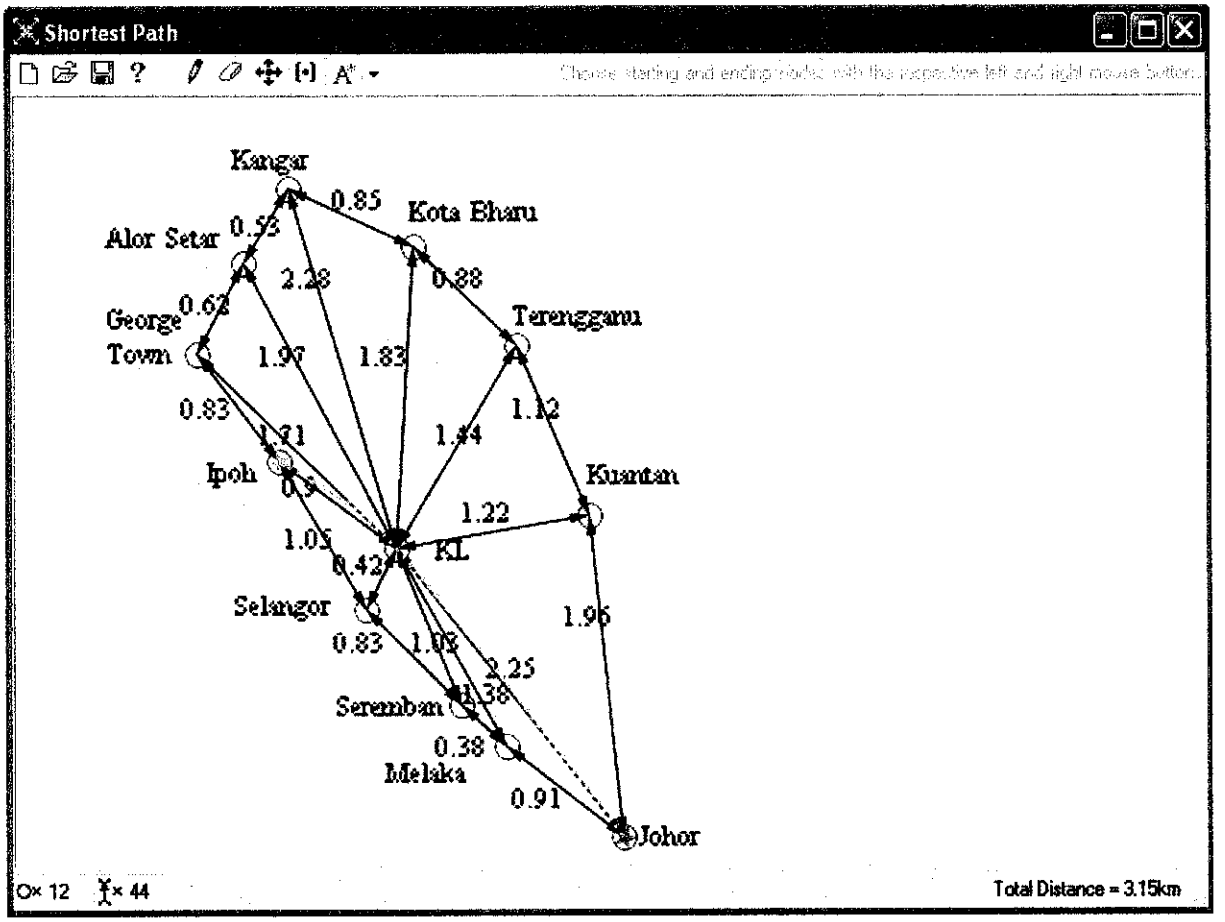


Figure 4.4: Test Case 2 (Malaysian State Road Network)

CHAPTER 5

CONCLUSION

The Automatic route finder for new visitors has been developed as the main purpose is to help new visitors to find the shortest path from the starting location to the destination. The system is simple and efficient in presenting visitors the visual presentation of the shortest path and distance. The automatic route finder has the ability to calculate shortest path distance quickly and display the location of the visitors with the highlighted path that the visitor is heading to. It will serve as a very useful tool for many visitors. Lot of time is saved when the shortest route finder is used. It is also important that the shortest route finder will not be more expensive when produced in large numbers and titled in all vehicles. The route finder designed can be implemented in larger road network for any selected city or any country.

Since route finder, by definition are intended to be used by the people and public, they must be easy to use for everyone. The system must have a uniform user interface where the method of entering, navigating in, and exiting the system should be consistent for the user. By having interactive route finder with virtual application might attract user to use the system. With additional functionality that required less time to calculate shortest path and also educate user with much information could increase the level of understanding. The system that being developed is simple, easy to understand and observable. These are the main goals that need to be achieved during the development of end product. The software worked perfectly and computed the shortest path very quickly, and will work on large and complicated networks. It is recommended that future work can be carried by developing a simple intelligent monitor system which will work by finger touch of selected nodes.

REFERENCES

[1] Andrew S. Tanenbaum, "Computer Networks" Prentice Hall, August 2002 fourth edition, chap 5.2 Routing Algorithms.

[2] J E Beasley, < http://people.brunel.ac.uk/~mastjjb/jeb/or/graph.html> OR-Notes.

[3] Chabini, I., 1997, A new algorithm for shortest paths in discrete dynamic networks, as presented at 8th IFAC/IFIP/IFORS Symposium on transportation systems, Tech Univ Crete, Greece, 16-18 June 1997.

[4] Husdal, J., 2000, an investigation into fastest path problems in dynamic transportation networks. Unpublished, coursework for the MSC in GIS, University of Leicester. <http://husdal.com/mscgis/research.htm>

[5] Collischonn, W. and Pilar, J.V., 2000, A direction dependent least-cost-path algorithm for roads and canals. International Journal of Geographic Information Systems, 12, 491-508.

[6] Roozbeh Razabi, 2001, How Routing Algorithms Works, <http://computer.howstuffworks.com/routing-algorithm1.htm>

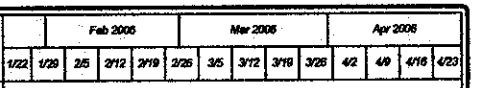
[7] Vijay Mukhi's, "Visual Studio.Net with C#" Tech Publications PTE LTD, 2002 First Edition, 35 - 157

[8] Dudley W. Gill, "Building Web Applications with C# and .NET" CRC Press 2003, .NET Framework pg 3 - 20

APPENDICES

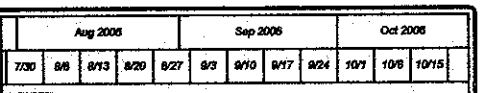
A) Gantt Chart for the first semester

ID	Task Name	Start	Finish	Duration
1	- Selection of Project Topic	1/23/2006	1/27/2006	1w
2	- Preliminary Research Work - Project Planning	1/30/2006	2/10/2006	2w
3	- Submission of Prelim Report	2/6/2006	2/17/2006	2w
4	- Project Work - Literature - Practical and Laboratory Work	2/13/2006	3/10/2006	4w
5	- Submission of Progress Report	3/13/2006	3/17/2006	1w
6	- Continue of Project Work - Practical and Laboratory Work	3/13/2006	4/7/2006	4w
7	- Submission of Interim Report Final Draft	4/10/2006	4/14/2006	1w
8	- Oral Presentation	4/17/2006	4/21/2006	1w
9	- Submission of Interim Report	4/24/2006	4/28/2006	1w

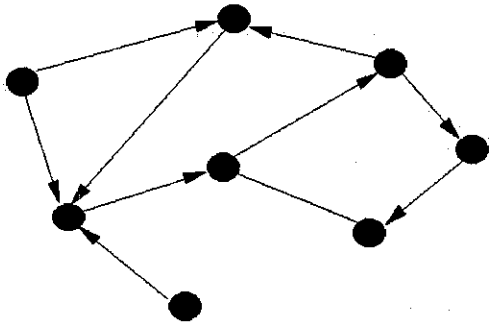


B) Gantt Chart for the second semester

ID	Task Name	Start	Finish	Duration
1	Project Work Continue	7/28/2006	8/7/2006	7d
2	Submission of Progress Report 1	8/7/2006	8/11/2006	5d
3	Project Work Continue	8/11/2006	9/8/2006	21d
4	Submission of Progress Report 2	9/8/2006	9/15/2006	6d
5	Project Work Continue	9/8/2006	10/6/2006	21d
6	Submission of Dissertation Final Draft	10/6/2006	10/13/2006	6d
7	Oral Presentation	10/13/2006	10/20/2006	6d
8	Submission of Project Dissertation	10/20/2006	10/27/2006	6d



C) Example of a graph



D) Source Code for Dijkstra's algorithm

```
#define MAX_NODES 1024       /* maximum number of nodes */
#define INFINITY 1000000000    /* a number larger than every maximum path */
int n,dist[MAX_NODES][MAX_NODES];   /*dist[I][j] is the distance from i to j */
void shortest_path(int s,int t,int path[ ])
{struct state {                 /* the path being worked on */
int predecessor ;               /*previous node */
int length                      /*length from source to this node*/
enum {permanent, tentative} label   /*label state*/
}state[MAX_NODES];
int I, k, min;
struct state *
            p;
for (p=&state[0];p < &state[n];p++){     /*initialize state*/
p->predecessor=-1
p->length=INFINITY
p->label=tentative;
}
state[t].length=0; state[t].label=permanent ;
k=t ;                                   /*k is the initial working node */
do{                                     /* is  the better path from k? */
for I=0; I < n; I++)                      /*this graph has n nodes */
if (dist[k][I] !=0 && state[I].label==tentative){
        if (state[k].length+dist[k][I] < state[I].length){
    state[I].predecessor=k;
    state[I].length=state[k].length + dist[k][I]
  }
```

```
}
/* Find the tentatively labeled node with the smallest label. */
k=0;min=INFINITY;
for (I=0;I < n;I++)
    if(state[I].label==tentative && state[I].length <
min)=state[I].length;
    k=I;
}
state[k].label=permanent
}while (k!=s);
/*Copy the path into output array*/
I=0;k=0
Do{path[I++]=k;k=state[k].predecessor;} while (k > =0);
}
```