QUESTION 1

Assuming you were to build a navigational system that routes vehicle through the shortest path to desired location. Perform a system analysis and outline the steps with the aid of a flow chart, required to achieve this.

SOLUTION

In order to build an efficient navigation system several steps need to be followed as depicted in the flowchart below:

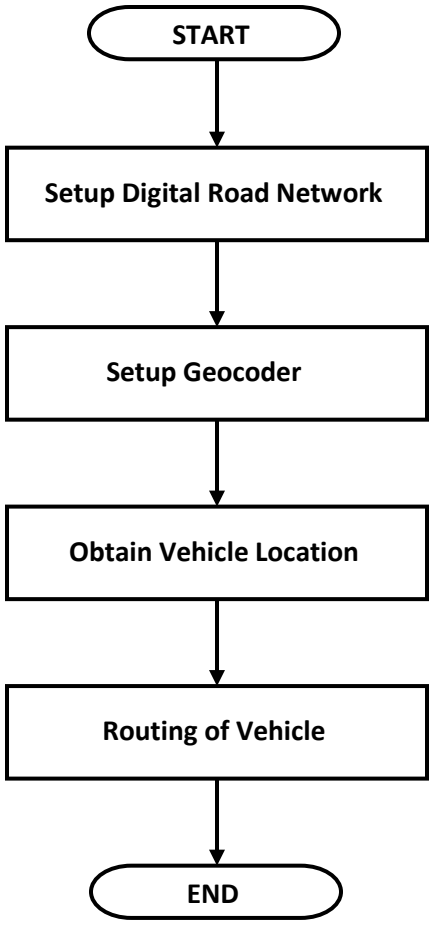

Figure 1: Navigation System Flowchart

The process to be carried out for each step is detailed below:

1.1 Setup Digital Road Network

The platform of an efficient navigational system is an accurate digital road map dataset, since it plays a major role in functions such as map-matching, datum transformation, geocoding and shortest-path search. Map accuracy is the relative precision in the position of an object on a map as compared to the real position of the object on the surface of the

earth. The accuracy of a digital map is not dependent on the scale of the map but on the accuracy of the source data, the transformation algorithms, and the resolution of the map. Also, slight corrections can be made to the source map utilizing ground truthing (actual ground measurements or coordinates obtained from a GPS).

The setup of the digital road network involves the creation of a spatial representation of the physical road network. This is usually accomplished by storing the related links (road segment) and nodes (start and end points of a road segment) in a spatial database. Each road segment is stored as line-string geometry and each node is stored as point geometry.

Digital road network maps could be obtained from various sources in several formats e.g. ESRI shapefile. Most spatial databases provide tools for importing these data into the database as a geometry format.

1.2 Setup Geocoder

Geocoding is the process of finding associated geographic coordinates from other geographic data such as street addresses, or zip/postal codes. The internal processing procedures that produce the geographic coordinates require complex interpolation algorithms such as standardization (token parsing and complex probabilistic models), normalization (structuring and refining of input addresses), and attribute relaxation (specifying matching priority for elements of an input address).

The setup of a Geocoder could be accomplished by utilizing the geocoding functionalities of some spatial databases such as Oracle. The Oracle spatial Geocoder is a robust tool for geocoding addresses utilizing a reference data set and in-built set of functions for processes such as address normalization and interpolation. Alternatively, there are available APIs which could be utilized for the geocoding of an address location. This takes away the complexity involved in building a Geocoder from scratch.

1.3 Obtain Vehicle Location

The geographic coordinates of the vehicle's location is obtained from a GPS device, datum transformation is utilized to convert the coordinates of the GPS device to that of the digital road network where necessary, and the accuracy of the GPS coordinates is improved by a process called map-matching. The map-matching process is explained overleaf:

1.3.1 Map-matching

Map-matching is the process of utilizing location data from a GPS device and the data of the spatial road network to provide an enhanced spatial reference for the GPS location. The overall purpose of a map-matching algorithm is to ascertain the actual road segment on which a vehicle is moving and to establish the vehicle's location on that segment, as illustrated in Figure 2Figure .

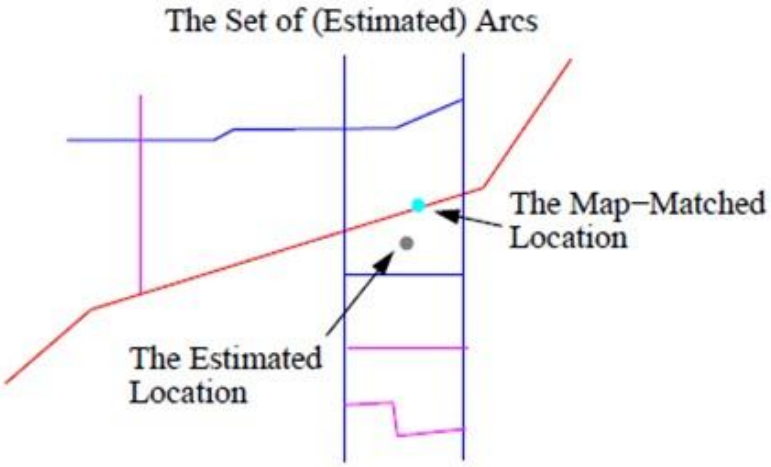

Figure 2: The Map-matching process

The accuracy of the road map used with the map-matching algorithm plays a major role in the performance of the algorithm. Also, during the map-matching process it is necessary to ensure that the coordinate system of the GPS location is the same as that of the underlying road network.

Over the years, several map-matching algorithms have been produced with improved accuracy due to the use of enhanced techniques, improvement in the functionality of positioning sensors, and the quality and magnitude of spatial road network data. These algorithms can be classified into two main categories – geometric map-matching, and topological/history data map-matching.

1.4 Routing of Vehicle

The coordinates obtained from the geocoding of an address location and the coordinates obtained by the map-matching of a vehicle's GPS location to the road network is utilized by a navigation system to determine the shortest-path from the vehicle to the location. These coordinates are dynamically inserted into the road network graph (link and nodes).

Some spatially enabled databases such as Oracle and PostGIS provide a network interface for determining the shortest-path between points on a road network and provision is

made by the network interface for constraints such as one-way road segments. The Oracle network interface utilizes the Dijkstra's algorithm to perform its shortest-path analysis.

An understanding of the Dijkstra's shortest-path algorithm is essential in a situation where the network interface provided by a spatial database is not going to be utilized.

1.4.1 The Dijkstra's Shortest-Path Algorithm

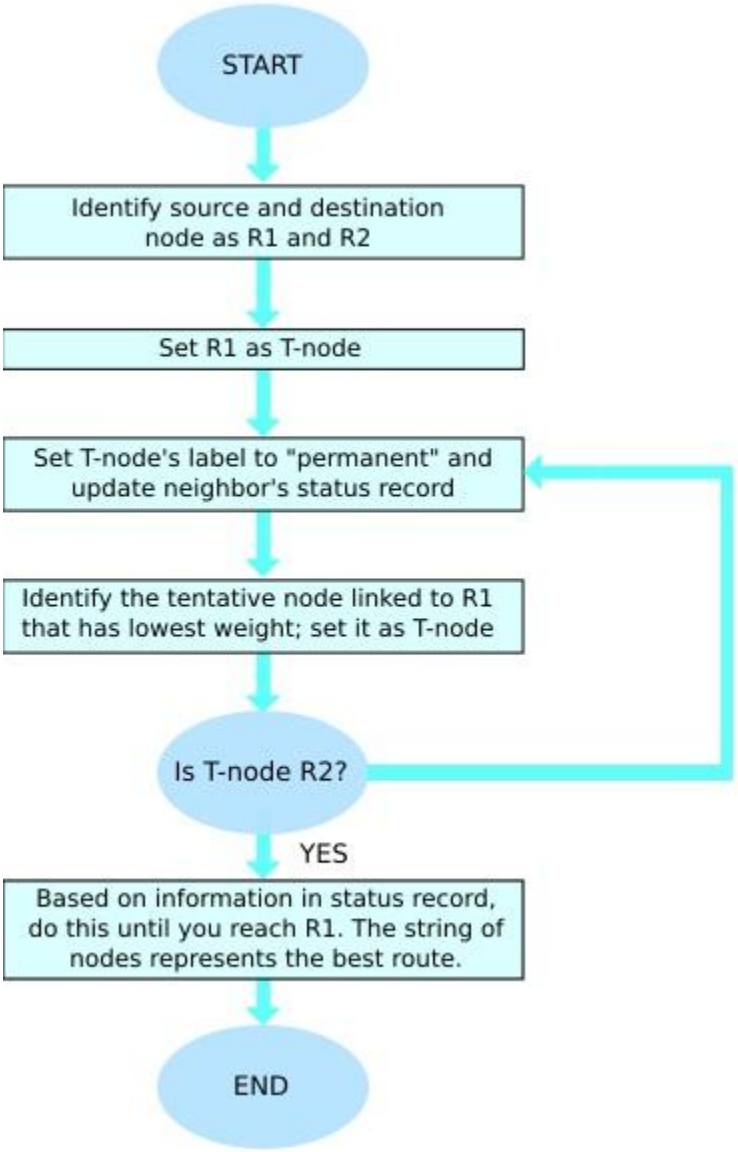

Figure 3: Flowchart Representation of Dijkstra's Shortest-Path Algorithm

The Dijkstra's Algorithm finds the shortest path from a source to all destinations in a directed graph. The flowchart in Figure 3 depicts the steps involved in the Dijkstra's shortest-path algorithm. The steps are detailed below:

1. The first step is to build a graph of the network. Then identify the source and destination nodes, for example R1 and R2. A matrix, called the "adjacency matrix" is

built. In the adjacent matrix, a coordinate indicates the weight, w, which would be the distance of a road segment in a navigation system. For instance, w[i, j] is the weight of a link between nodes Ri and Rj. If there is no direct link between Ri and Rj, this weight is identified as "infinity."

2. The next step is to build a status record for each node on the network. The record contains the following fields:
    - Predecessor field - shows the previous node.
    - Length field - shows the sum of the weights from the source to that node.
    - Label field - shows the status of node; each node have one status mode: "permanent" or "tentative."
3. The next step is to initialize the parameters of the status record (for all nodes) and set their label to "tentative" and their length to "infinity".
4. The next step is to set a T-node. If R1 is to be the source T-node, for example, the label of R1 is changed to "permanent." Once a label is changed to "permanent," it never changes again.
5. The status record for all tentative nodes that are directly linked to the source T-node is updated.
6. The next step is to go over all tentative nodes and choose the one whose weight to R1 is lowest. That node is then the destination T-node.
7. If the new T-node is not R2 (the intended destination), go back to step 5.
8. If this node is R2, the router extracts its previous node from the status record and does this until it arrives at R1. This list of nodes shows the best route from R1 to R2.

There are several implementations of the Dijkstra's algorithm. A java implementation could be obtained from the link below.

http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html

REFERENCES

Communication Networks/Routing. Retrieved August 13, 2016, from https://en.wikibooks.org/wiki/Communication_Networks/Routing

Vogel L. (2011). *Dijkstra's shortest path algorithm in Java.* Retrieved August 13, 2016, from http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html