A Seminar Report on:

“Application Of Graph Theory In Online Distance Mapping Applications”

Prepared by: Shubham Kamleshbhai Bhadada

Roll. No. : U19CS028

Class : B.Tech –III (Computer Science & Engineering) 5th Semester

Year : 2021-22

Guided by : Dr . Sankita J. Patel

Department of Computer Engineering
Sardar Vallabhbhai National Institute of Technology,
Surat -395007 (Gujarat), India

Sardar Vallabhbhai National Institute of Technology,
Surat -395007 (Gujarat), India

CERTIFICATE

This is to certify that the seminar report entitled **Application of Graph Theory in online distance mapping Applications** is prepared and presented by **Mr. Shubham K. Bhadada** bearing Roll No.: **U19CS028**, 3th Year of **B. Tech (Computer Science and Engineering)** and his work is satisfactory.

GUIDE	JURY	HOD
Dr. Sankita J. Patel		COED

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

A*	A-Star
API	Application Programming Interface
DAG	Directed Acyclic Graph
GPS	Global Positioning System

Abstract

The main transportation problem in a city is finding the shortest paths to reach the destination. All of the cities nowadays are experiencing high traffic. This study is aimed to implement the algorithms for searching the shortest route. There are many algorithms available for finding the shortest path. In this research, I tried to understand online distance mapping applications and their functionalities. I have used multiple algorithms to find the shortest route for real-life data and then compared them based on their results and cost.

The calculations have used the concept of the graph Theory. The supporting map was obtained from Google Maps; The best route was calculated using the algorithms like Dijkstra's, A-Star and Bellman-Ford. The results from these algorithms were analyzed and compared. All three algorithms successfully found the best route.

Chapter 1

Introduction

Online Distance mapping applications are those packages meant to provide the consumers with multiple routes that they can traverse to move from source to destination.The apps use GPS; since most mobile phones are well equipped with GPS, they use satellite signals to accurately determine the consumer/user's location.

However, these apps predict the location of the nearby landmarks, and then they try to predict your exact location.The most important feature of these categories of apps is finding the shortest route between source and destination, i.e., let us say you want to go from Ahmadabad to Delhi; these apps help users find the fastest time to reach Delhi. These apps will consider multiple factors such as: Traffic , Size of the Routes , The Maximum Speed of the Vehicle , Maximum Speed that can be achieved on the route.

To enable the user to reach the destination in the shortest time, these Applications have various API that use the graph theory to calculate the most optimal path for the user to traverse. To be precise, API use graph Algorithms such as:

- Dijkstra's Algorithm
- A* Algorithm
- Bellman-Ford Algorithm

The approximate map between the source and the destination is converted into a graph. Then the above algorithms are implemented to find the shortest path between the origin and destination.

Thus, these apps help save a large number of resources, such as time, money, etc., for the users and are proved to be much more effective in multiple navigation purposes.

1.1 Motivation

The motivation behind choosing ”Application of Graph Theory in Online distance mapping Applications ” as a research topic lies behind the fact that technology has kept changing for the better and mankind has reached to a point where it is finally able to achieve results significantly closer to the ones obtained by human brain.

Graph Theory has provided researchers and developers with a new domain of research, and it can be explored even further than the current findings. It required a lot of patience and persistence to reach the current position in this particular field.

The way Graph Theory has revolutionised the navigation system is something worth studying in a detailed manner. This has been my source of motivation.

1.2 Objectives

The primary objective of this Seminar Report is to highlight the study and research based on the topics:

- Graph Theory Algorithms.
- Shortest Path Problem.
- Implement the Algorithms in real life Situation

1.3 Organisation Of Project

The First chapter gives a basic idea and objective of the report. Second Chapter Presents Background regarding the Graph Theory. Third Chapter enriches reader with the features of Online distance mapping Applications. Fourth Chapter discusses various representations of any locality to a Graph. Fifth Chapter makes reader aware about various factors affecting time taken to move from source to destination. Sixth Chapter discusses various algorithm used by these platforms along with their pseudo-codes. Seventh Chapter consists of the real-time Implementation of above algorithms along with results and analysis.

Chapter 2

Background

Graph theory is the branch of mathematics that is concerned with networks of points connected by lines. The subject of graph theory had its origins in exciting math problems. However, it has grown into a vital area of mathematical research, with applications in:

- Operations research
- Chemistry
- Anthropology
- Computer science.

The history of graph theory could also be drawn explicitly to the mid-18th century when the Switzerland based mathematician Euler had solved the Konigsberg bridge problem. The Konigsberg bridge problem was an old problem concerning the likelihood of finding a route over each one of seven bridges that traverse a branched river flowing past an island but without going over any bridge twice.[1][7][13]

Euler claimed that no such way exists. His proof covered only understandings of the links' physical arrangement, but still, he proved the primary theorem in graph theory.[1]

Graph theory uses the expression graph to not refer to data charts, such as line graphs or bar graphs. Instead, it suggests a set of vertices or points or nodes and edges or lines that connect the vertices.

2.1 Types Of Graph[1]

Multigraph

When more than one edge joins any two vertices of the Graph, then the graph is called a multi-graph.

Simple Graph

A simple, strict graph is an unweighted and undirected graph that does not contain multiple edges between two vertices and does not contain any loops. It may be either connected or disconnected.

Except asserted differently, the graph is assumed to refer to a simple graph. When an edge connects each vertex to every other vertex, the graph is called a complete graph.

Di-Graph

Whenever we add any direction to the edges, the graph created is a directed graph or Di-graph i.e. when we assign direction to the edges of the graph, if a Di-graph does not contain any cycle it is called Directed Acyclic Graph or DAG.

Tree

Tree is the special form of graph which has following properties :

- It is Undirected .
- It is Connected.
- It does not contain any cycles or loops.

2.2 Terminology

Degree

A significant thing related to each vertex is its degree, defined because the number of edges entering or exiting it. Thus, a loop adds 2 to the degree of its vertex. We can comment on its fundamental nature if we know about the number of vertices in a complete graph. For this

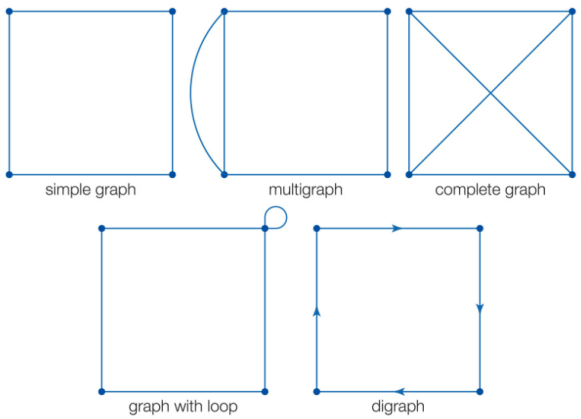

Figure 2.1: Different Types of Graph[1]

reason, we denote complete graphs as K_n, where n refers to the number of vertices. All vertices of K_n have each vertex connected to every other vertex.

So, Euler's theorem about the Konigsberg Bridge problem can be summed up into the following points:

- If there is any path along the edges of a multigraph that traverses each edge only once, then there are at most two vertices whose degree is odd.
- If the path begins and ends at the same vertex, then there won't be any vertices with odd degrees.

Path

Another essential notion in graph theory is the path, which is a way(route) through the edges of a graph. A path may consist of a single edge directly between two vertices or multiple edges spanning through various vertices.

For every pair of vertices in the graph, If there is a path between them, the graph is connected.

A path that originates and ends at the same vertex and encounters every edge exactly once is called a circuit or closed path.

A circuit that follows each edge once while visiting every vertex is an Eulerian circuit, and therefore the graph is named an Eulerian one. An Eulerian graph has the following properties:

- It is connected.
- All of its vertices have even degrees.

Chapter 3

Features[5]

In today's lifestyles, time and money are critical; nobody desires to waste their money and time travelling on a long course instead of a quick direction. Or it may be giving more money for public transportation as you were unaware of the delivery fee.

Online distance mapping packages were brilliant for travellers as they provided customers with the path they wanted to journey. The consumer can find the course from source to destination spot or even manually using GPS to reach their purpose. They also help customers pick out different kinds of transportation, i.e., bus, train, vehicle, and so forth.Online Distance mapping Apps are essentially internet-based navigation structures. They consist of comprehensive, precise maps in different countries and territories.

Following are few features of Distance Mapping applications:

- It allows the customers to search for distinctive places around them and also provides some information about the other places which the user wants.They also allow the user to find their locations on maps by GPS.
- They are used for getting locations of another city. The user can get directions for another area concerning his location.
- They give different options to the user to select their mode of transportation, i.e., bus, train and walking.They also give the distance and time for travelling from one place to another to the user.
- They help the user by providing directions during driving, public transportation and walking paths for several towns and cities,the Live traffic conditions, incident reports, automatic rerouting to find the best route , Street Views , Satellite views and etc.

Chapter 4

Factors affecting Traversal time

Now we will discuss various factors that affect the cost and duration of the travel for traversing from source to destination. Below are some of the elements:

- Distance: The total distance between the source and destination is the most defining criteria to evaluate the duration of traversal from the start to the goal.
- Traffic: The second most important criterion after the distance between source and destination, the greater is the traffic, the greater is the duration of the journey; .The traffic may be caused due to various reasons such as: Accidents , Disabled Vehicles , Malfunctioning Traffic Signals , etc.
- Familiarity with Route: The most obvious, one determines the degree of ease with which a person reaches the destination. The more the person is familiar with the route lesser is the time to get to the destination.
- Road Quality and Capacity: The Quality and Capacity of the Road is also a significant criterion to decide the fate of the journey.
- Speed Limit: The maximum speed limit that can be attained during the journey is also one of the criteria.
- Toll-Booth Charges: Let us say two routes, A and B, take the same amount of time to reach from source to destination, but while traversing path A , toll booth encounters us hence we need to pay more money in route A. Hence we can say that route B is more optimal than route A,But this factor is not considered by many Applications.

Chapter 5

Representation of localities in the form of Graph

Any particular locality can be easily represented as the spatial networks in maps, but we need a better mathematical structure to calculate distance. Graphs are those structures that fill the void here. We can construct the Graph from a set of nodes and edges, assigned to different components in the map.[3][14]

Here the spatial network can be represented as a planar undirected Graph; We will describe our network in the standard notation, i.e. $G(V, E)$, where V is the set of the vertices while E is the set of the edges.

Related to graph G, we will also consider a few technical terms such as Adjacency Matrix, Incidence Matrix and Degree.

The Adjacency Matrix represented as $A(G) = (a_{ij})_{i,j=1}^{n}$ of the Graph G is a square matrix with size $n \times n$, the value of a_{ij} = 1 if there is an edge from vertex i to vertex j, and a_{ij} = 0 when there isn't any edge between them , with all the diagonal entries of the matrix as 0.

The Incidence Matrix represented as $B(G) = b_{ij}$, for $i = 1, 2, ..n$ and $j = 1, 2, \ldots m$ of graph G is matrix with size $n \times m$ such that $b_{ij} = 1$, if edge e_j and vertex v_i are incident and 0 otherwise. The Degree Matrix represented as $D(G) = d_{ii}$ for $i = 1, 2, ..n$ is a diagonal matrix whose elements are the degree of the nodes.[7][8][13][14]

Here we will discuss a few of the spatial networks which are used to generalize the maps to Graph:

5.1 Primal Graph[3]

It is the earliest model of the spatial network; it is generalized into a planar undirected Graph. Leonard Euler proposed it in mid of sixteenth century. The roads or the streets are represented as the edges of the Graph. The intersection points of the roads or streets are the vertices of the Graph constructed. The significant advantage of the primal Graph is that they accurately describe the spatial and topographical structure of the locality.

We can also derive a relation between the Adjacency matrix, Incidence Matrix and Degree Matrix as follows:$A(G) = B(G)B(G)^T$–$D(G)$.

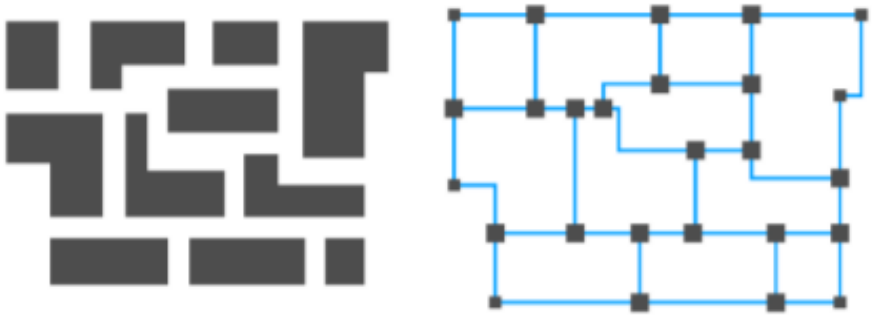

Figure 5.1: Conversion of urban space to Primal Graph[3]

5.2 Dual Graph[3]

- It is another network used to define spatial networks in a locality.
- Dual Graphs, in general terms, can be defined as the transition from one concept to another in one to one manner by an involution task.
- Dual Graphs are further divided into their different subcategories:
  - Edges to Vertex Dual Graph: The streets are generalized into the nodes, and the intersections of the roads are the edges. They use linearity of streets as the criteria for the continuity
  - Face to Vertex Dual Graph: The nodes are generalized as primal graph faces while faces are represented as nodes of the primal one.

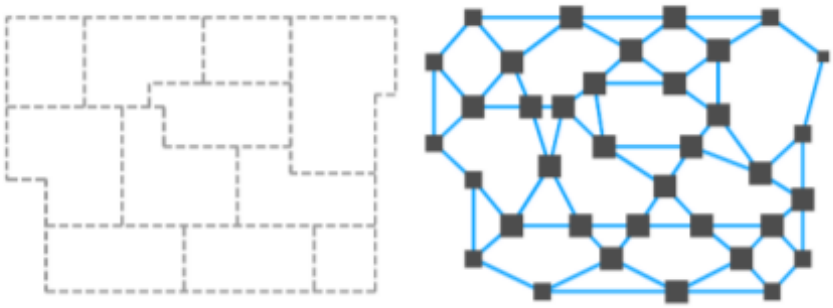

Figure 5.2: Conversion of urban space to Edge to Vertex Dual Graph[3]

5.3 Delaunay Graph[3]

- This type of network serves a different purpose than the previous ones since in these types of networks arrangements, the principal aim here is not to represent the topology of the layout but to establish spatial relationships between the components of the environment.

5.4 Features of each Representation

Hence we have seen different types of graph arrangements to depict the network. It has also been seen that the primal Graph is the most suitable one to represent the topology and geometry of the network. But it does not help us to understand characteristics of network morphology, To do so, we use dual graphs to represent the network. We can use Delaunay Graph to study dis-positional relationships amongst the faces of the graphs.

Chapter 6

Algorithms

Here we will discuss the following algorithms used by different online distance mapping applications:

- Dijkstra's Algorithm
- A* Algorithm
- Bellman-Ford Algorithm

We will use a primal representation of the locality, which we discussed earlier, i.e. all the streets are represented as edges, and the intersection of roads are the vertices.[15][17][19]

6.1 Dijkstra's Algorithm

One of the famous apps, Google Maps, is based on the above Algorithm, which Edsger W. Dijkstra invented. He was a Dutch system scientist, programmer, software developer and a pioneer in computer science.

Dijkstra algorithm is a greedy algorithm. It uses a paradigm that builds the answer piece by piece, always choosing the most optimum amount at every point of choice; unlike this, not every greedy Algorithm presents the most optimal result.

Dijkstra's Algorithm, also known as Dijkstra's Shortest Path First algorithm, is a greedy algorithm for finding the shortest paths between nodes in a graph that may represent, for example, computer networks or roads. This Algorithm exists in multiple variants such as:

1. The shortest path between two nodes.

2. The shortest route from the source node to all other nodes.

It will be primarily be used for edges with only positive weights.

Now we will see the pseudo-code for Dijkstra's algorithm, which will have the following parameters:

- $Graph$ be a Graph containing details about all the vertices and the edges of the graph.
- Priority Queue Q with each of its elements as pair of (weight, vertex) , where weight is weight of respective vertex.
- $dist[]$ is the array with the size of number of vertices where the i^{th} element in the array denotes the distance from Source and $par[]$ is the array with size of number of vertices where the i^{th} element in array denotes parent of it during the path source to destination.
- Let $Source$ be source vertex. and Let d be the destination vertex.

Algorithm 1 Dijkstra's Algorithm

$Q.push((0,0))$
for all v in Graph **do**
  $dist[v] \leftarrow \infty$
end for
$dist[source] \leftarrow 0$
while Q is not empty **do**
  $temp \leftarrow Q.top()$
  $Q.pop()$
  for all neighbour u of temp.vertex **do**
    $alter \leftarrow length(u,v) + temp.weight$
    if $alter < dist[u]$ **then**
      $dist[u] \leftarrow alter$
      $par[u] \leftarrow temp.vertex$
      $Q.push((alter,u))$
    end if
  end for
end while

6.2 A* Algorithm

A* (A-star) is a graph traversal and path finding algorithm widely used in multiple computing fields due to its efficiency and optimality. Hence it is commonly used in Multiple Online distance mapping applications and is still the best in numerous test cases.

It is an extension of Dijkstra's Algorithm. A* achieves optimal results by using heuristics to guide its search.

A* is a best-first search, meaning that it is generalized in terms of weighted graphs; it aims to find the shortest route from the source to the required destination node.

At each step, A* determines which node is nearest to the current node, i.e. which of its path is to be created using a function:

- $F(n) = G(n) + H(n)$

Where n is the next node, $G(n)$ is the cost from source to node, and $H(n)$ is a heuristic function that estimates the cost of the cheapest path from n to goal. The procedure is terminated when we reach the destination.

One interesting fact is that Dijkstra's Algorithm is the particular case of A* algorithm where $H(n) = 0$.

Now we will see the pseudo-code for A* algorithm, which will have the following parameters:

- $Graph$ be a Graph containing details about all the vertices and the edges of the graph.
- Priority Queue Q with each of its elements as pair of (weight, vertex) , where weight is weight of respective vertex.
- $dist[]$ is the array with the size of several vertices where the i^{th} element in the array denotes the distance from Source and $par[]$ is the array with size of number of vertices where the i^{th} element in array denotes parent of it during the path source to destination.
- Let $Source$ be source vertex and d be the destination vertex.
- $F(v) = dist[v] + h(v, d)$, here $h(v, d)$ is the heuristic function defined by $h(v, d) = ((v_x - d_x)^2 + (v_y - d_y)^2)^{0.5}$, here we are assuming that our heuristic function is Euler's Distance.

Algorithm 2 A* Algorithm

$Q.push((F(0), 0))$
for all v in Graph **do**
$dist[v] \leftarrow \infty$
end for
$dist[source] \leftarrow 0$
while Q is not empty **do**
$temp \leftarrow Q.top()$
$Q.pop()$
for all neighbour u of temp.vertex **do**
$alter \leftarrow length(u, v) + dist[temp.vertex]$
if $alter < dist[u]$ **then**
$dist[u] \leftarrow alter$
$par[u] \leftarrow temp.vertex$
$F(u) \leftarrow dist[u] + h(u, d)$
$Q.push((F(u), u))$
end if
end for
end while

6.3 Bellman-Ford Algorithm

We will now see the solution of the above problem, i.e. the single-source shortest path problem within the general case where we will allow negative weights within the graph.One might think how negative weights of the edges make sense. They probably don't talk about distances on the map, but other factors reduce problems to the shortest paths, and negative weights show up in these reductions.

Here we note that if a negative weight cycle (the sum of weights on the cycle is negative) can be reachable from the source, there does not exist any solution. Every time we go around the cycle, we get a path with a shorter length, so to find the quickest route, we would go around for eternity.

The Algorithm initializes all the distance of the source to 0 and all other nodes to ∞. Then

for all the possible edges, if the distance to the endpoint can be minimized by taking the edge, the distance is updated to the new smaller value. At each step i that the edges are scanned, the Algorithm finds all shortest paths of at most length i edges of that number of steps.

Now we will see the pseudo-code for Bellman-Ford algorithm, which will have the following parameters:

- $Graph$ be a Graph containing details about all the vertices and the edges of the graph.
- $dist[]$ is the array with the size of several vertices where the i^{th} element in the array denotes the distance from Source and $par[]$ is the array with size of number of vertices where the i^{th} element in array denotes parent of it during the path source to destination.
- Let $Source$ be source vertex and d be the destination vertex.

Algorithm 3 Bellman-Ford Algorithm

for all v in Graph **do**
  $dist[v] \leftarrow \infty$
end for
$dist[source] \leftarrow 0$
for all u in Graph **do**
  for all edge (u,v) in Graph **do**
    $temp \leftarrow dist[u] + length(u, v)$
    if $temp < dist[v]$ **then**
      $par[u] \leftarrow temp.vertex$
      $dist[v] \leftarrow temp$
    end if
  end for
end for
for all edge (u,v) in Graph **do**
  if $dist[u] \leftarrow length(u, v) + dist[v]$ **then**
    ERROR: NEGATIVE CYCLE EXISTS
  end if
end for

Chapter 7

Real-life Implementation

7.1 Creation Of Graph

Now we will see and discuss the implementation of the above-discussed algorithms : For experimental purposes, I took an example of my district, and I tried to navigate from my town Sanand to a nearby village, Chaloda using all three algorithms. Let's take a look at the Satellite image of the required area.

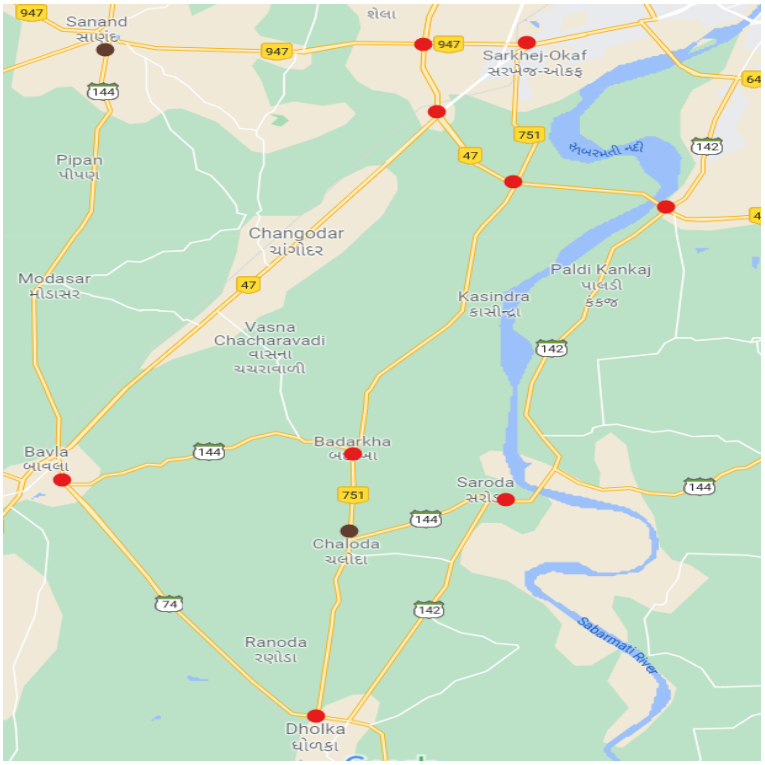

Figure 7.1: An image of locality from Google Maps

Here as mentioned earlier, the Source is Sanand, and the Destination is Chaloda, both marked in Brown Color. While all other nodes are marked by red colour. Now we will see the graphical representation of the above map to visualize connections in a better manner.

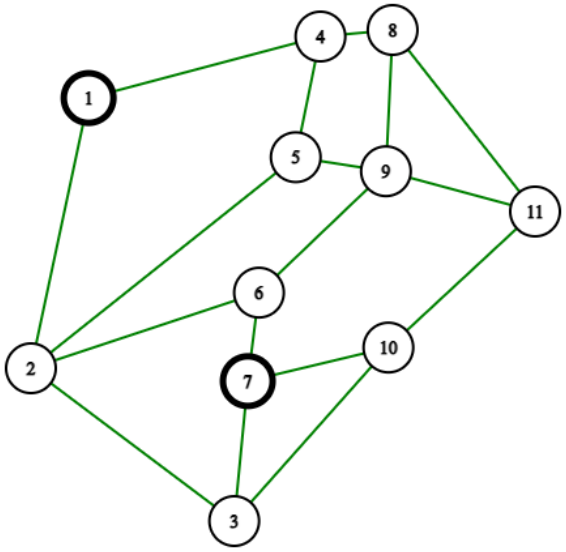

Figure 7.2: Graphical Representation of locality

Table 7.1: Details about Vertices of the graph containing the name of the locality which they represent along with heuristic function of respective vertex

Vertex ID	Locality Represented	Value of Heuristic Function in km
1	Sanand	21.09
2	Bavla	8.94
3	Dholka	7.21
4	Sahyog Hospital	20
5	Sanathal Circle	17
6	Badarkha	3.36
7	Chaloda	0
8	Sarkhej Police Station	20.5
9	Bakrol Police Chowki	15
10	Saroda	4.45
11	Kamod	16.17

Table 7.2: Details about edges of the graph containing Distance and Time required to traverse through respective edges

Edge Between Vertices	Distance between edges	Time to Traverse the edge
$1 \leftrightarrow 4$	14	24
$1 \leftrightarrow 2$	20	32
$2 \leftrightarrow 3$	14	22
$3 \leftrightarrow 7$	9	13
$3 \leftrightarrow 10$	12	19
$7 \leftrightarrow 6$	4.3	6
$6 \leftrightarrow 9$	12	16
$5 \leftrightarrow 9$	3.7	6
$5 \leftrightarrow 2$	20	22
$2 \leftrightarrow 6$	11	23
$4 \leftrightarrow 8$	7	15
$8 \leftrightarrow 9$	6.3	11
$8 \leftrightarrow 11$	13	23
$11 \leftrightarrow 10$	16	21
$10 \leftrightarrow 7$	4.3	6
$4 \leftrightarrow 5$	7	16
$9 \leftrightarrow 11$	5	9

7.2 Observations

After implementing above algorithms on our local system We have come to following results :

Table 7.3: Readings when we have tried to minimise Distance

Algorithm	Path	Distance(km)	Time(min)
Dijkstra's	$1 \rightarrow 2 \rightarrow 6 \rightarrow 7$	35.3	61
A-Star	$1 \rightarrow 2 \rightarrow 6 \rightarrow 7$	35.3	61
Bellman-Ford	$1 \rightarrow 2 \rightarrow 6 \rightarrow 7$	35.3	61

Table 7.4: Readings when we have tried to minimise Time

Algorithm	Path	Distance(km)	Time(min)
Dijkstra's	$1 \rightarrow 2 \rightarrow 6 \rightarrow 7$	35.3	61
A-Star	$1 \rightarrow 2 \rightarrow 6 \rightarrow 7$	35.3	61
Bellman-Ford	$1 \rightarrow 2 \rightarrow 6 \rightarrow 7$	35.3	61

7.3 Result and Analysis

Hence we have found that out of all possible paths the path $1 \rightarrow 2 \rightarrow 6 \rightarrow 7$ is the most optimal path to travel from from 1 to 7, Hence in order to travel from Sanand to Chaloda the path which would suit best to us is $Sanand \rightarrow Bavla \rightarrow Badarkha \rightarrow Chaloda$. After performing the analysis, I have tried to sum up efficiency of various algorithms[12][18][19] , Their use depends on the nature of the problem for which they are implemented .

Table 7.5: Comparison of the Algorithms

Algorithm	Negative Edges	Single Source	Time Complexity	Space Complexity
Dijkstra's	✗	✓	$O(\|E\| + \|V\|log\|V\|)$	$O(\|E\| + \|V\|)$
A-Star	✗	✓	Function of Heuristic	Function of Heuristic
Bellman-Ford	✓	✓	$O(\|V\|.\|E\|)$	$O(\|V\|^2)$

Chapter 8

Conclusion

8.1 Conclusion

The shortest path calculation has been widely implemented for GPS based applications to help users find the best path. Dijkstra's, A-Star, and Bellman-Ford algorithms have shown good results in finding the shortest path. Dijkstra algorithm performed better than the other two to find the best path concerning the algorithm's time complexity, but all the three algorithms produced the same result for the most optimal route. We can also say that Dijkstra's algorithm uses a greedy paradigm, while A* algorithm uses Heuristic Paradigm and Bellman-Ford Algorithm uses Dynamic Programming Paradigm to find the shortest path.

8.2 Future Prospects

The study can be enhanced with the dynamic route calculation when weight changes its value by a re-routing mechanism in the next research.

The study can also be enhanced by taking the help of Deep learning and Machine learning algorithms to find the most optimal Heuristic function for any particular case while we are using the A* algorithm.

Bibliography

[1] Carlson, Stephan C.. ”graph theory”. Encyclopedia Britannica, 24 Nov. 2020, https://www.britannica.com/topic/graph-theory. Accessed 15 October 2021.

[2] Natarajan, B. and Balaji, M.K.. (2019). Application of Graph Theory in Online Network Services to Determine the Shortest Journey. International Journal of Advanced Networking and Applications. 10. 4030-4034. 10.35444/IJANA.2019.10058.

[3] WIT Transactions on Engineering Sciences. 2017, 118: 71-82.

[4] Kikelomo, Akinwol and Nureni, Yekini and Paul, Adelokun and Olawale, Lawal. (2017). Design and Implementation of Mobile Map Application for Finding Shortest Direction between Two Pair Locations Using Shortest Path Algorithm : A Case Study. International Journal of Advanced Networking and Applications. 9. 3300-3305.

[5] Prof. Suvarna Pansambal, Nikhil Iyer , Vira Meherkar , Pankaj Sharma: Navigation Systems for Fastest Route. International Journal of Advanced Research in Computer and Communication Engineering,2016,DOI: 10.17148/IJARCCE.2016.53120 .

[6] Rodríguez-Puente, R., Lazo-Cortés, M.S. Algorithm for shortest path search in Geographic Information Systems by using reduced graphs. SpringerPlus 2, 291 (2013).

[7] Narasingh Deo, “Graph theory with applications to engineering and computer science”, Prentice Hall ofIndia, 1990

[8] Anandhan, Prathik & Uma, K. & Anuradha, J.. (2016). An overview of application of graph theory. 9. 242-248.

[9] K.S. Avdhesh, and K. Sourabh, Finding of Shortest Path from Source to Destination by traversing every node in wired Network, International Journal of Engineering and Technology, 5, 2013, 2655-2656.

[10] Pooja Singal and R.s.chhillar. Article: Dijkstra Shortest Path Algorithm using Global Position System. International Journal of Computer Applications 101(6):12-18, September 2014.

[11] Mehta, Heeket and Kanani, Pratik and Lande, Priya. (2019). Google Maps. International Journal of Computer Applications. 178. 41-46. 10.5120/ijca2019918791.

[12] Eneh, A. H. and Uchechukwu Christian Arinze. "COMPARATIVE ANALYSIS AND IMPLEMENTATION OF DIJKSTRA'S SHORTEST PATH ALGORITHM FOR EMERGENCY RESPONSE AND LOGISTIC PLANNING." Nigerian Journal of Technology 36 (2017): 876-888.

[13] Graph Theory with Applications (J. A. Bondy and U. S. R. Murty).

[14] Alain l'Hostis. Graph theory and representation of distances : chronomaps, and other representations. Mathis Ph. Graphs and networks, multilevel modelling,, Lavoisier, pp.177-191, 2007.

[15] R. Pramudita, H. Heryanto, R. Trias Handayanto, D. Setiyadi, R. W. Arifin and N. Safitri, "Shortest Path Calculation Algorithms for Geographic Information Systems," 2019 Fourth International Conference on Informatics and Computing (ICIC), 2019, pp. 1-5, doi: 10.1109/ICIC47613.2019.8985871.

[16] F. Riaz and K. M. Ali, "Applications of Graph Theory in Computer Science," 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, 2011, pp. 142-145, doi: 10.1109/CICSyN.2011.40.

[17] Helshani, Laurik. "An Android Application to solve the shortest path problem using Google Services and Dijkstra's Algorithm." (2016).

[18] Abusalim, Samah & Ibrahim, Rosziati & Saringat, Mohd & Jamel, Sapiee & Wahab, Jahari. (2020). Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization. IOP Conference Series: Materials Science and Engineering. 917. 012077. 10.1088/1757-899X/917/1/012077.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd. ed.). The MIT Press.

Acknowledgment

The opportunity to thoroughly study the concept of ”Application Of Graph Theory in online distance mapping Applications” was undoubtedly worthwhile, presenting gratitude towards my mentor, Dr . Sankita J. Patel, who guided me on every step of preparing the seminar report. Her contribution to helping me create an ideal Seminar Report and presenting my findings is unparalleled.

I would also like to thank Dr . Udai P. Rao for providing all the required guidelines and necessities to prepare a well-curated seminar report and constant supervision.

I would also like to thank Dr . Mukesh A. Zaveri, HOD, Computer Engineering Department, for supervising the course.