Bangladesh Journal of Multidisciplinary Scientific Research; Vol. 1, No. 2
ISSN 2687-850X E-ISSN 2687-8518
July-September; 2019
Published by Centre for Research on Islamic Banking & Finance and Business

Finding Shortest Path for Road Network Using Dijkstra's Algorithm

Md. Almash Alam
Lecturer
Department of Computer Science and Engineering
Bangladesh Army University of Engineering & Technology, Natore, Bangladesh
E-mail: almash.anik.61@gmail.com

Md. Omar Faruq
Lecturer
Department of Computer Science and Engineering
Bangladesh Army University of Engineering & Technology, Natore, Bangladesh
E-mail: omarfarukcse10@gmail.com

Abstract
Roads play a Major role to the people live in various states, cities, town and villages, from each and every day they travel to work, to schools, to business meetings, and to transport their goods. Even in this modern era whole world used roads, remain one of the most useful mediums used most frequently for transportation and travel. The manipulation of shortest paths between various locations appears to be a major problem in the road networks. The large range of applications and product was introduced to solve or overcome the difficulties by developing different shortest path algorithms. Even now the problem still exists to find the shortest path for road networks. Shortest Path problems are inevitable in road network applications such as city emergency handling and drive guiding system. Basic concepts of network analysis in connection with traffic issues are explored. The traffic condition among a city changes from time to time and there are usually huge amounts of requests occur, it needs to find the solution quickly. The above problems can be rectified through shortest paths by using the Dijkstra's Algorithm. The main objective is the low cost of the implementation. The shortest path problem is to find a path between two vertices (nodes) on a given graph, such that the sum of the weights on its constituent edges is minimized. This problem has been intensively investigated over years, due to its extensive applications in graph theory, artificial intelligence, computer network and the design of transportation systems. The classic Dijkstra's algorithm was designed to solve the single source shortest path problem for a static graph. It works starting from the source node and calculating the shortest path on the whole network. Noting that an upper bound of the distance between two nodes can be evaluated in advance on the given transportation network.

Keywords: Shortest Path; Dijkstra's Algorithm; Breadth First Search; Maximum Number of Nodes.

1. Introduction
This paper involved in illustrating the best way to travel between two points and in doing so, the shortest path algorithm was created. Dijkstra's Algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and other network related protocols. For a given source vertex (node) in the graph, the algorithm finds the S finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

A large amount of work has been done on finding shortest paths through abstract. Dijkstra's algorithm is a Shortest Path Finding Algorithm which is applicable on a Graph which is Directed and got the Edges weighted with non-negative weights. If we have an undirected graph with edges unweighted, we can solve the problem with the implementation of Breadth First Search (BFS) algorithm. As we know we are unable to use BFS algorithm in case of a weighted and directed graph, but we can modify our algorithm of BFS in such a way that we can handle the above stated issues (weights and direction). We will later know that Dijkstra's algorithm is asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded nonnegative weights. These are the motivations those will help us to know about the Dijkstra's algorithm more in future.

2. Related Works

The indoor navigational assistance for this type of users presents additional challenges not faced by convention alguidance systems, due to the personal nature of the interactions. The algorithms are part of an overall Indoor Navigation Model that is used to provide assistance and guidance in unfamiliar indoor environments. Path planning guses the Dijkstra's shortest path algorithms, to operate on an "Intelligent Map", that is based on a new data structure termed "cactus tree" which is predicated on the relationships between the different objects that represent an indoor environment. This research explores the potential to design an application for the visually impaired even when to- date' positioning and tracking' system cannot offer reliable position information that highly required by this type of application. The best-path problem for public transportation systems, we found that the nature of transfer is that it requires extra costs from an edge to its adjacent edge. So,we use the direct/indirect adjacent edges weighted directed graph as a public transportation data model and improve the storage of an adjacency matrix. We introduce the spaces to rage structure in order to store the scattered information of transfer in indirect adjacent edges lists. Thus, we solve the problem of complex network graphs' storage and design a new shortest path algorithm to solve transit problem based on the data model. Algorithm analysis exhibits that the data model and the algorithm we propose are prior to a simple graph based on the Dijkstra's algorithm in terms of time and space**.** Today, the increased traffic and complex modern road network have made finding a good route from one location to another a non-trivial task. Many search algorithms have been proposed to solve the problem, and the most well-known being Dijkstra's algorithm, Johnson's algorithm. While these algorithms are effective for path finding, they are wasteful in terms of computation. In this paper, we present a study to examine both uninformed search and heuristic search based on some major cities and towns. Efficient usage of routing algorithms can significantly reduce travelling distance and transportation costs as well. Usage of the shortest-path algorithm in this case Dijkstra's algorithm for inner transportation optimization in warehouses. The model integrates data such as: 2D graph of warehouse layout, its 3D extension, historical data, ABC analysis and business process model. The prototype of proposed application is tested with sample data and by simulating different working conditions.

3. Graph Theory

A graph G is a collection of two sets V & Ewhere V is the collection of vertices v0,v1,................Vn-1 also called nodes and E is thecollection of edges e1, e2,...............en where anedge is an area which connects two nodes. This can be represented as – G=(V,E)
V(G)=(v0,v1,.........vn)or set of vertices

E(G)=(e1,e2,..........en)or set of edges

4. Overview of Shortest Path Algorithm

There are several cases in graph where we have a need to know the shortest path from one node to another node. General electric supply system and water distribution system also follow this approach. The best example we can take is of a railway track system. Suppose one person wants to go from one station to another then he needs to know the shortest path from one station to another here station to another. Here station represents the node and tracks represent edges. In computers, it is very useful in network for routing concepts. There can be several paths for going from one to another node. But the shortest path is that path in which the sum of weights o the included edges is the minimum.

4.1 Dijkstra's Algorithm

Let the node at which we are starting be called the initial node. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step. For the first iteration, the current intersection will be the starting point and the distance to it will be zero. For subsequent iterations (after the first), the current intersection will be the closest unvisited intersection to the starting point—this will be easy to find.From the current intersection, update the distance to every unvisited intersection that is directly connected to it. This is done by determining the sum of the distance between an unvisited intersection and the value of the current inters section and relabeling the unvisited intersection with this value if it is less than its current value. In effect, the intersection is relabelled if the path to it through the current intersection is shorter than the previously known paths. To facilitate shortest path identification, in pencil, mark the road with an arrow pointing to the relabeled intersection if you label/relabel it, and erase all others pointing to it. After you have updated the distances to each neighbouring intersection, mark the current intersection as *visited* and select the unvisited intersection with lowest distance (from the starting point) – or the lowest label—as the current intersection. Nodes marked as visited are labelled with the shortest path from the starting point to it and will not be revisited or returned to.

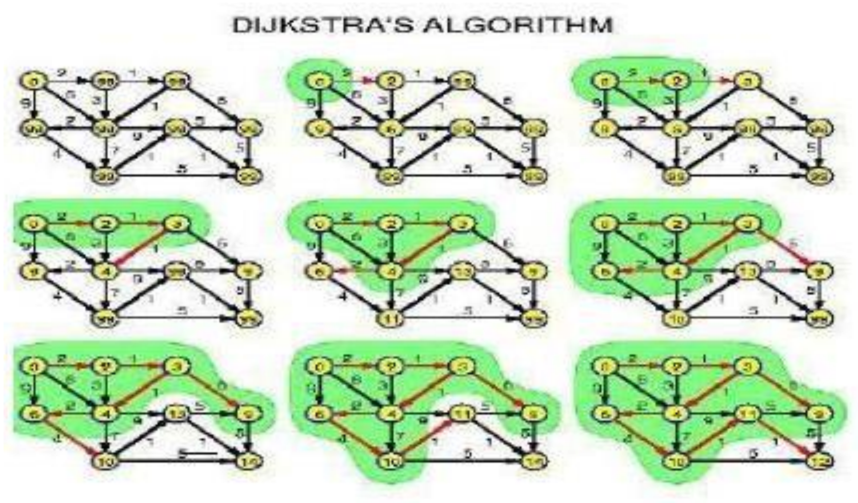

Fig. 1: Example

4.2 Algorithm

// direction line between p1 and p2 dx = node[p2].x-node[p1].x; dy = node[p2].y-node[p1].y;
// distance between p1 and p2
l = (float)(Math.sqrt((float)(dx*dx + dy*dy))); dir_x[p1][p2]=dx/l; dir_y[p1][p2]=dy/l;
// calculate the start and endpoints of the arrow,
// adjust startpoints if there also is an arrow from p2 to p1 if (weight[p2][p1]>0) { startp[p1][p2] = new Point((int)(node[p1].x-5*dir_y[p1][p2]),
(int)(node[p1].y+5*dir_x[p1][p2]));
endp[p1][p2] = new Point((int)(node[p2].x-5*dir_y[p1][p2]),
(int)(node[p2].y+5*dir_x[p1][p2]));
}
else {
startp[p1][p2] = new Point(node[p1].x, node[p1].y); endp[p1][p2] = new Point(node[p2].x, node[p2].y);
}
// range for arrowhead is not all the way to the start/endpoints intdiff_x = (int)(Math.abs(20*dir_x[p1][p2]));
intdiff_y = (int)(Math.abs(20*dir_y[p1][p2]));
// calculate new x-position arrowhead if
(startp[p1][p2].x>endp[p1][p2].x) { arrow[p1][p2] = new Point(endp[p1][p2].x + diff_x + (Math.abs(endp[p1][p2].x-startp[p1][p2].x) - 2*diff_x)* (100-w)/100 ,0);
}
else {
arrow[p1][p2] = new Point(startp[p1][p2].x + diff_x + (Math.abs(endp[p1][p2].x-startp[p1][p2].x) - 2*diff_x)*

w/100, 0);
}
// calculate new y-position arrowhead if (startp[p1][p2].y>endp[p1][p2].y) {
arrow[p1][p2].y=endp[p1][p2].y + diff_y + (Math.abs(endp[p1][p2].y-startp[p1][p2].y) - 2*diff_y)* (100-w)/100;
}
else {
arrow[p1][p2].y=startp[p1][p2].y + diff_y + (Math.abs(endp[p1][p2].y-startp[p1][p2].y) - 2*diff_y)* w/100;

5. Experimental Results

In this section, we conduct an experiment to compare the paths between the nodes. In that paths, the shortest path was done by using dijkstra's algorithm. Here the shortest path it means low cost was found by the shortest path algorithm. Here java code uses the applets so that applet viewer shows how the shortest path was done. With the use of java software the result was shown in the report. We have got successful result. The experiment was successfully complete. The result of this project finding the shortest path for a single source to all pairs of vertices by using the Dijkstra's Algorithm. It gives the cheapest cost and its implementation is easy.

5.1 Initial

Algorithm running: Red arrows point to nodes reachable from the start node.
The distance to: b=55, d=50, f=100. Node d has the minimum distance. There is no other arrows coming in to d. Node d will be colored orange to indicate 50 is the length of the shortest path to d.

5.2 Final

Orange arrows point to nodes reachable from nodes that already have a final distance. The distance to: c=68.There are no other arrows coming in to c.
Node c will be colored orange to indicate 68 is the length of the shortest path to c. Algorithm has finished, follow orange arrows from start node to any node to get the shortest path to the node.

6. Conclusion

We have proposed a practical algorithm for the shortest path problem in transportation networks. The proposed algorithm can limit the search in a sub-graph based on the given nodes of the distance between the two nodes. As a B result, the calculation for the shortest path has been simplified. Experimental results on a real-world road network reflect the potential characteristic of the proposed algorithm in comparison to the existing works. B Applets can prove to be very helpful to all the students, in general. Every student who wishes to learn one of the algorithms, implemented in this tool, may use the proposed applet from any remote place, probably his home. It doesn't matter what his system is. It only requires that the Java Virtual Machine, (JVM), is installed into his system. It should be pointed out that JVM is freely available for download from Sun Microsystems. Moreover there is an intention to enhance the graph editor, in order that the user would be able to draw undirected graphs, or networks, as well. Yet this applet could be further enriched, and possibly visualize more algorithms for digraphs. Directed networks such as the Primal Simplex Algorithm which solves the Minimum Cost Network Flow Problem (MCNFP). Besides more algorithms will be added for undirected graphs such as the Kruskal's and Prim's algorithms which solve the Minimum Spanning Tree (MST) problem.

Future Work

In future release the applet will be modified, in order to be executed as standalone Java application as well. In addition, certain questionnaires will be given to the students of our department to be filled in, in order to get feedback. This is very important, to improve and evaluate the teaching effectiveness of our applet in real instruction environment. Students will also be encouraged to construct their own applets.

References

Aghaei, M. R. S., Zukarnain, Z. A., Mamat, A., & Zainuddin, H. (2008, December). A Hybrid Algorithm for the Shortest-path Problem in the Graph. In *2008 International Conference on Advanced Computer Theory and Engineering* (pp. 251-255). IEEE.

Abbas, M. A., Chumachenko, S. V., Hahanova, A. V., Gorobets, A. A., & Priymak, A. (2013, September). Models for quality analysis of computer structures. In *East-West Design & Test Symposium (EWDTS 2013)* (pp. 1-6). IEEE.

Alzahrani, A. S., & Woodward, M. E. (2008, November). End-to-end delay in localized qos routing. In *2008 11th IEEE Singapore International Conference on Communication Systems* (pp. 1700-1706). IEEE.

Dobrilovic, D., Jevtic, V., Beker, I., & Stojanov, Z. (2012, May). Shortest-path based model for warehouse inner transportation optimization. In *2012 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*(pp. 63-68). IEEE.

Fuhao, Z., & Jiping, L. (2009, August). An algorithm of shortest path based on Dijkstra for huge data. In *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery* (Vol. 4, pp. 244-247). IEEE.

Guo, L., Yang, Q., & Yan, W. (2012, September). Intelligent path planning for automated guided vehicles system based on topological map. In *2012 IEEE Conference on Control, Systems & Industrial Informatics* (pp. 69-74). IEEE.

Jan, G. E., Lee, M. C., Hsieh, S. G., & Chen, Y. Y. (2009, July). Transportation network navigation with turn penalties. In *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (pp. 1224-1229). IEEE.

Jigang, W., Han, P., Jagadeesh, G. R., & Srikanthan, T. (2010). Practical algorithm for shortest path on large networks with time-dependent edge-length. In *2010 2nd International Conference on Computer Engineering and Technology*.

Jain, A., Datta, U., & Joshi, N. (2016). Implemented modification in Dijkstra‟ s Algorithm to find the shortest path for N nodes with constraint. *International Journal of Scientific Engineering and Applied Science*, *2*(2), 420-426

Wu, H., Marshall, A., & Yu, W. (2007, July). Path planning and following algorithms in an indoor navigation model for visually impaired. In *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)* (pp. 38-38). IEEE.

Wang, H., Hu, M., & Xiao, W. (2010, March). A new public transportation data model and shortest-path algorithms. In *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)* (Vol. 1, pp. 456-459). IEEE.

Xiao, J. X., & Lu, F. L. (2010, February). An improvement of the shortest path algorithm based on Dijkstra algorithm. In *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)* (Vol. 2, pp. 383-385). IEEE.

Yin, C., & Wang, H. (2010, June). Developed Dijkstra shortest path search algorithm and simulation. In *2010 International Conference on Computer Design and Applications* (Vol. 1, pp. V1-116). IEEE.

Zhang, Z., Jigang, W., & Duan, X. (2010, December). Practical algorithm for shortest path on transportation network. In *2010 International Conference on Computer and Information Application* (pp. 48-51). IEEE.

Zhan, F. B., & Noon, C. E. (1998). Shortest path algorithms: an evaluation using real road networks. *Transportation science*, *32*(1), 65-73.

Zhou, W., Qiu, Q., Luo, P., & Fang, P. (2013, June). An improved shortest path algorithm based on orientation rectangle for restricted searching area. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 692-697). IEEE.