International Journal of Engineering & Technology, 7 (4.10) (2018) 974-976

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Dijkstra Algorithm Application: Shortest Distance between Buildings

G. Deepa[1], Priyank Kumar[2], A. Manimaran[3*], K. Rajakumar[4], V. Krishnamoorthy [5]

[1,3]*School of Advanced Sciences, VIT, Vellore – 632014, India*
[2,4]*School of Computer Science and Engineering, VIT, Vellore-632014, India*
[5]*College of Natural Sciences, Arba Minch University, Ethiopia*
**Corresponding author E-mail: deepa.g@vit.ac.in*

Abstract

The shortest path algorithm is one of the best choices for implementation of data structures. The shortest path (SP) problem involves the problem of finding a suitable path between “two vertices or nodes in a graph” in such a way that the sum of the weights of its component edges is minimal. There are many theories for solving this problem one of the widely used way solution for solving this problem is Dijkstra’s algorithm (DA) which is also widely used in many engineering calculation works also. There are two types of DA one is the basic one and other one is optimized. This paper is focused on the basics one which provides a shortest route between source node and the destination node. Main focus has been kept on keeping the work simple and easy to understand with some basic concepts .Storage space and operational efficiency improvement has been tried to improve.

Keywords: *directed graph (DG); DA; SP*

1. Introduction

The SP problem determines a path which is having a minimum weight connecting two given specified vertices one of them is considered as source and other one as destination (or goal) in a weighted graph also known as DG (digraph) (see [1]). A simple and very good algorithm used to find the SP between two given vertices in a DG is DA. DA has increased time and space complexity (See [3], [4], [5] and [6]). This paper has proposed the simple way of applying SP algorithm for the calculation of SP among the buildings present in the campus.

2. Preliminary works

In this section we discussed the graphs with “lengths/weights/costs on edges”, SP in edge-weighted graphs and classic DA for computing single-source SP.

2.1. Dijkstra algorithm

(See [2]) Given a graph with all vertices select any vertex as a source vertex and then find SP. First, we have to generate a “SP tree (SPT) with the source as root”. We will have to maintain “two sets first set contains vertices included in SPT and other set includes vertices not yet included in SPT”. In every step we have to find a vertex which is the latter set and possess minimum distance from source. This algorithm solves the SP problem under constrain that there should be no negative weight cycle present in the graph. If there is any negative weight cycle then SP will not be detected.

2.2. Graphs with edge “length”

(See [7]) An edge-weighted directed graph, $G=(V,E,\omega)$ has a Length/weight/cost function $\omega : E \to N$ which maps each edge $(u,v)\in E$ to a non-negative integer "length" (or "weight", or "cost"): $\omega(u,v)\in N$. We can extend the "length" function ω to a function $\omega : V\times V \to N\cup\{\infty\}$, by letting $\omega(u,v)=0$, for all $u\in V$, and letting $\omega(u,v)=\infty$ for all $(u,v)\notin E$.

2.3. Dijkstra’s single-source shortest-path algorithm

Input: Edge-weighted graph, $G=(V,E,\omega)$ with (extended) weight function $\omega : V\times V \to N$, and a source vertex $s\in V$.
Output: Function $L : V \to N\cup\{\infty\}$, such that for all $v\in V$, $L(v)$ is the length of the shortest path from s to v in G.
Algorithm:

```
Initialize: $S:=\{s\}; L(s):=0;$
Initialize: $L(v):=\omega(s,v)$, for all $v\in V-\{s\}$;
while $(S\neq V)$ do
$u:=\arg\min_{z\in V-S}\{L(z)\}$
$S:=S\cup\{u\}$
for all $v\in V-S$ such that $(u,v)\in E$ do
$L(v):=\min\{L(v),L(u)+\omega(u,v)\}$
end for
end while
Output function $L(.)$.
```

3. Pseudo code

```
1. Create a cost matrix ct$[V][V]$ from adjacency matrix graph$[V][V]$.
2. Array visited$[\,]$ is initialized with value $0$
3. If vertex $3$ is source vertex then visited $[3]$ is marked $1$
4. Create a distance matrix $n$ by sorting the cost of vertices from vertex $0$ to $n-1$ from source vertex $3$
5. Choose a vertex $x$ such that distance $[x]$ is minimum and visited $[x]$ is $0$ mark visited $[x]$ as $1$
6. Again calculate the shortest distance of remaining vertices from the source.
```

4. Procedure

The main idea behind this paper is to implement this algorithm in our college campus to find shortest path between buildings. For the weights between the edges we took the coordinates of all the building through google map. Using these coordinates distance between all the building were calculated through a small code which is given below. By having an idea of the positions of all the buildings, a rough graph is created. From the rough graph adjacency matrix is created now through online tool for the creation of undirected graph through adjacency matrix a proper weighted graph is created.

(See [8]) The "haversine formula to calculate the great-circle distance between two points that is, the shortest distance over the earth's surface" is given by

$$Longitude = long2 - long1$$

$$Latitude = lat2 - lat1$$

$$X = \left(\sin^2\left(latitude/2\right)\right) + \cos\left(lat1\right) * \cos\left(lat2\right) * \left(\sin^2\left(longitude/2\right)\right)$$

$$Y = 2 * a\tan 2\left(\sqrt{X}, \sqrt{1-X}\right)$$

$$Z = R * Y$$

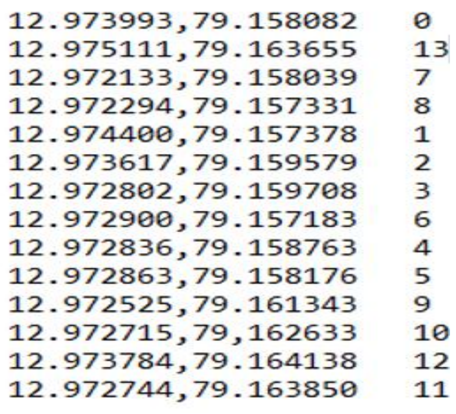
Where R is radius of earth.

5. Numerical example

Example application built using dev $c++$ in c language. Our aim is to find the shortest path to reach any building from a start building.

The program takes input as the starting node and it calculates the shortest distance between the start nodes to all the nodes present in the graph.

Here we coordinates of various building represented as numbers from 0 to 13.



12.973993,79.158082	0
12.975111,79.163655	13
12.972133,79.158039	7
12.972294,79.157331	8
12.974400,79.157378	1
12.973617,79.159579	2
12.972802,79.159708	3
12.972900,79.157183	6
12.972836,79.158763	4
12.972863,79.158176	5
12.972525,79.161343	9
12.972715,79,162633	10
12.973784,79.164138	12
12.972744,79.163850	11

Fig.1: Coordinate of various building represented as numbers from 0 to 13

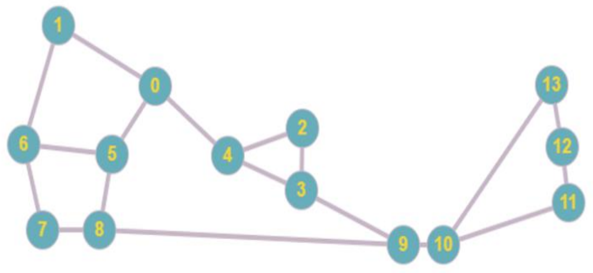


Fig. 2: Rough graph

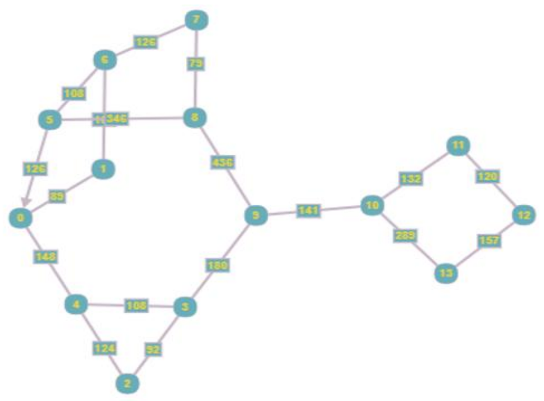
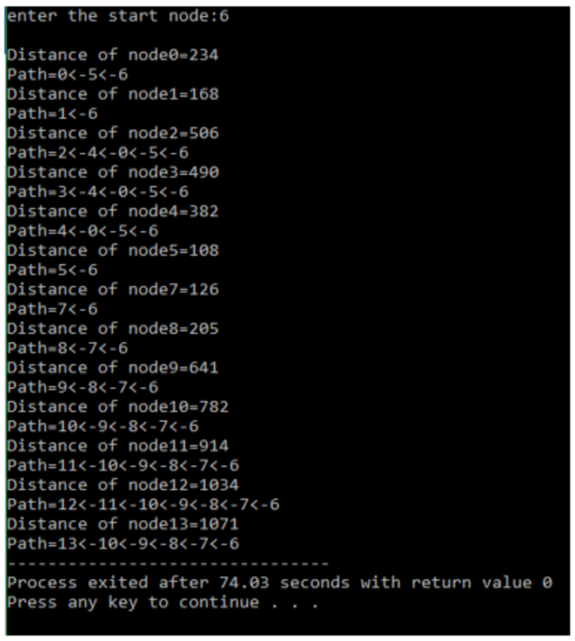


Fig. 3: Undirected weighted Graph

```
enter the start node:6
```

Fig. 4: Output window showing node 6 taken as input



```
enter the start node:6

Distance of node0=234
Path=0<-5<-6
Distance of node1=168
Path=1<-6
Distance of node2=506
Path=2<-4<-0<-5<-6
Distance of node3=490
Path=3<-4<-0<-5<-6
Distance of node4=382
Path=4<-0<-5<-6
Distance of node5=108
Path=5<-6
Distance of node7=126
Path=7<-6
Distance of node8=205
Path=8<-7<-6
Distance of node9=641
Path=9<-8<-7<-6
Distance of node10=782
Path=10<-9<-8<-7<-6
Distance of node11=914
Path=11<-10<-9<-8<-7<-6
Distance of node12=1034
Path=12<-11<-10<-9<-8<-7<-6
Distance of node13=1071
Path=13<-10<-9<-8<-7<-6
--------------------------------
Process exited after 74.03 seconds with return value 0
Press any key to continue . . .
```

Fig. 5: Output window showing shortest path of node 6 to all other nodes

6. Conclusion

The above paper is a simple application of shortest path algorithm in our daily life. This paper presents the way how anyone can apply Dijkstra algorithm to get shortest path to reach the destination of any area or locality in which they live if they follow the process mentioned in the paper. The reader of this paper should have some basic understanding of programming language *c* through which they can implement the above pseudocode in a working code.

Acknowledgement

The authors thank the managements for their constant support towards the successful completion of this work.

References

[1] B.S. Hasan, M.A. Khamees and A.S.H. Mahmoud (2007), A Heuristic Genetic Algorithm for the Single Source Shortest Path Problem, *Proc. of International Conference on Computer Systems and Applications*, pp.187-194.

[2] J. Chamero (2006), Dijkstra's Algorithm Discrete Structures and Algorithms.

[3] T. Li, L. Qi, and D. Ruan (2008), An Efficient Algorithm for the Single-Source Shortest Path Problem in Graph Theory, *Proc. of 3rd International Conference on Intelligent System and Knowledge Engineering*, Vol. 1 , pp. 152-157.

[4] YAN Han-Bing LIU Ying-Chun (2000), A New Algorithm for Finding Shortcut in a City's Road Net Based on GIS Technology, *Chinese Journal of Computers*, Vol. 23, No.2, pp. 210-215.

[5] Arun Kumar Sangaiah, Minghao Han and Suzi Zhang (2014), An Investigation of Dijkstra and Floyd Algorithms in National City Traffic Advisory Procedures, *IJCSMC*, Vol. 3, No. 2, pp. 124-138.

[6] F. Benjamin Zhan (1997), Three fastest shortest path algorithms on real road networks: Data structures and procedures, *Journal of geographic information and decision analysis*, Vol. 1, No.1, pp. 69-82.

[7] https://www.inf.ed.ac.uk/teaching/courses/dmmr/slides/17-18/lec-shortest-coloring.pdf.

[8] https://andrew.hedges.name/experiments/haversine/.