Dijkstra SSSP Algorithm: Foundation of the Genius Google Maps

Dicky Novanto 13515134[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13515134@std.stei.itb.ac.id

***Abstract*—As the information technology in our world has been improving greatly, people tend to use the technology provided to them. One of the well-known technology is Global Positioning System (GPS). GPS has been included in many mobile applications, and one of them is the most famous among the civilians, that is Google Maps. This applications is very beneficial for knowing the rout of a trip that is not familiar to the user. Google Maps, certainly, uses graph as a tool that commonly used by computer scientists and Dijkstra algorithm as a basic algorithm to determine the shortest path –and probably the fastest path for users- between a place to another place (Single-Source Shortest Path or commonly known as SSSP). This paper will discuss firstly about the basic theorem about graph and Dijkstra SSSP algorithm, then the application of graph, Dijkstra algorithm as a basic tool for Google Maps doing its features, and also the modified Dijkstra algorithm to fasten the process of pathfinding.**

***Keywords*—Google Maps, Dijkstra SSSP algorithm, shortest path, graph, source, destination.**

I. Introduction

Nowadays, many people are using Global Positioning System (GPS) technology, for knowing the exact location of their places and also some other places. But using only GPS is not enough for fulfilling people's needs as the needs are continuously growing, for example, people need to know a path, from a place to a certain destination that is not familiar for them. The job cannot only be done using GPS because GPS is only giving a location of a certain place. Of course, knowing only the position of the users and the position of the destination the users want to go is not the answer. Therefore, there are many pathfinding program that has been published and commonly used by millions of users, and one of them is Google Maps.

Of course path that is desired is not the random path, which is very annoyingly far for going to a certain location, but the shortest path from a location to another location, which is also hoped as a fastest route. Therefore, there must be an algorithm for finding a shortest path to a destination, and one of the famous algorithm is Dijkstra SSSP (Single-Source Shortest Path) algorithm for computing the shortest path from a single source to another destinations. A detailed explanation about this algorithm will be covered in the third chapter of this paper.

II. Basic Theorem of Graph

[RIN06]Graph is a pair of sets (V, E) which consist of:
V = a non-empty set of vertices = {v_1 , v_2 , v_3 , … , v_n}
and
E = a set of edges connecting a pair of vertices = {e_1 , e_2 , e_3 , … , e_n}

According to the edges' orientation, graph can be classified into undirected graph and directed graph. Here will be defined directed graph as writer will be discussing more about directed graph. Directed graph is a graph containing edges that each of the edges has orientation [RIN06]. As the edges has a direction from a vertex (v_1) to another vertex (v_2), it forms an edge (v_1, v_2). v_1 is called an initial vertex and v_2 is called a terminal vertex and it is different from edge (v_2, v_1).

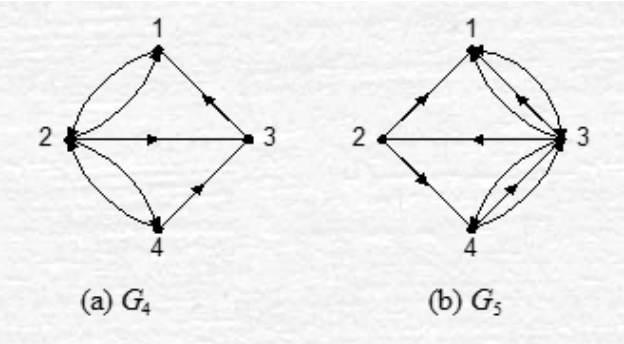

Figure 1: G_4 is a directed graph without multiple edges, and G5 is directed graph with multiple edges.
Source: Rinaldi Munir's Graf (2015) Slide

Graph is also classified based on the existence of the edges' weight. Those are unweighted graph and weighted graph. Unweighted graph is a graph with all identical edges' weight, hence, the weight of the edges in unweighted graph is not written along with the edges. In contrast, weighted graph is a graph that each edge can

have a different value or same value. The weight of each edge is usually written along with each edges in drawing the weighted graph.

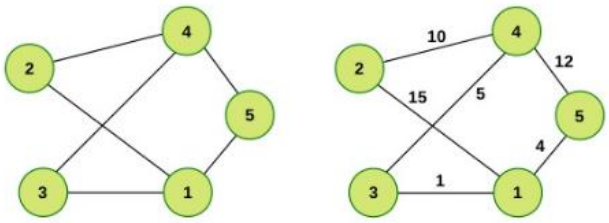

Figure 2: (From left to right) Unweighted graph and weighted graph
Source: http://www.slideshare.net/emersonferr/20-intro-graphs accessed in December 6, 2016 at 11.37 p.m.

III. DIJKSTRA ALGORITHM

A. *History and Algorithm explanation*

Dijkstra SSSP Algorithm was invented by Edsger Wybe Dijkstra in 1956 and he published it in 1959. [RIN06]This algorithm is basically using greedy paradigm as in each step, it takes a node that has a minimum weight and has never been visited before.

As we are going to deal with map and its paths, then the graph we are going to discuss is directed and weighted graph.

The Dijkstra SSSP Algorithm is as follow:

1. Set a certain initial node and set the tentative value of distance from an initial node to the other nodes (including the initial node). Setting the distance form initial state to itself as 0 and infinity to the other nodes besides the initial node.
2. Set all of the node as unvisited node and the initial node as a current node.
3. From current node, add the value of the current node and the edge's weight of all of its unvisited neighbor node. For instance, if the value of the current node is 4 (total of the shortest distance from the initial node), and the weight of the edge connecting the current node to the unvisited neighbor is 6, then the value of its unvisited neighbor is 10.
4. The sum of the addition is compared with the tentative distance of the node and the tentative distance of the node is replaced with the minimum of the sum result and the tentative distance.
5. When finished considering all of its neighbor, mark the current node as visited node. Visited node will never be checked again.
6. If there are no unvisited node remaining or the smallest tentative distance of the unvisited node is infinity, which means that the unvisited node is unreachable from the initial node, then the algorithm has finished.
7. If the algorithm has not finished, continue the algorithm by selecting an unvisited node with smallest tentative distance then do the step 3 again until the algorithm finishes.

B. *Finding the shortest path of a certain graph*

Here will be illustrated how the Dijkstra SSSP algorithm works. Consider this directed and weighted graph:

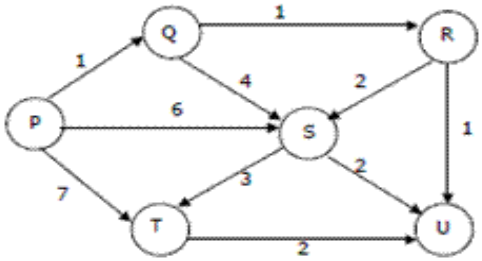

Figure 3: Directed and weighted graph
Source: http://quiz.geeksforgeeks.org/algorithms/graph-shortest-paths/ accessed in December 8, 2016 at 09.56 p.m.

Here node P is set as the initial node of the graph and the shortest path to any node different from P (as distance from P to P is automatically 0) will be found. Following the steps of the Dijkstra SSSP algorithm, the first step is:

From Node	Q	R	S	T	U
P	INF	INF	INF	INF	INF

Table 1: the first step of the Dijkstra SSSP algorithm
Edited using Microsoft Office Excel 2013

From the table, it can be inferred that from node P, the tentative distance to the other nodes (Q, R, S, T, U) is infinity.

From Node	Q	R	S	T	U
P	INF	INF	INF	INF	INF
P	1	INF	6	7	INF

Table 2: The second step of the algorithm

The unvisited neighbor node is Q, S, and T, and the weight of edge from P to Q is 1, from P to S is 6, and from P to T is 7. The minimum of the 3 edges' weight is 1 so node Q is chosen and marked as the visited node. Node P as the initial node is marked as the visited node as well.

From Node	Q	R	S	T	U
P	INF	INF	INF	INF	INF
P	1	INF	6	7	INF
Q	1	2	5	7	INF

Table 3: The third step of Dijkstra SSSP algorithm relating to the graph

From the table 3, we can see that the tentative distance of R is 2 at the moment because it is the sum of the current value of Q, that is 1, and the weight of edge from node Q to node R is 1. Taking a look at the node S, it shows that the tentative value of node S has reduced from 6 to 5. It is owing to the edge's weight from node Q to S is 4, hence the tentative value is 1+4 that is 5. The minimum value of 5 and 6 (from the previous data) is 5, therefore the tentative distance to node S is changed to 5. The tentative value of T and U remains the same since they are not reachable from node Q.

The minimum tentative distance of 2, 5, 7, and infinity is surely 2, so node R is chosen as the next node and marked as the visited node. Note that value 1 of the node Q is not considered as node Q is a visited node.

After a thorough analysis in each node, we finally get the shortest path from the initial node P to the other nodes, as shown:

From Node	Q	R	S	T	U
P	INF	INF	INF	INF	INF
P	1	INF	6	7	INF
Q	1	2	5	7	INF
R	1	2	4	7	3
U	1	2	4	7	3
S	1	2	4	7	3
T	1	2	4	7	3

Table 4: The final computation of the algorithm

The table shows that (especially the final row which is marked yellow) the numbers represent the shortest path from node P to others node, that is the shortest path from:

1. P to Q is 1
2. P to R is 2
3. P to S is 4
4. P to T is 7, and
5. P to U is 3

C. Complexity

Here will be explained the worst case time complexity using big-Oh notation of the Dijkstra SSSP algorithm applied in directed and weighted graph.

Assuming that the graph has V vertices and E edges. If the program uses priority queues that is already implemented in library of a certain programming languages, e.g. C++, initially, all vertices are pushed into the priority queue. The time complexity when it comes to pop a certain vertex from the priority queue is O(log V) using heap implementation. Then, the total time complexity to pop all the vertices is O(V.log V).

Inside the algorithm, edges that are connected to a certain vertex are also used as the computation of shortest path to each node. When a function that handle the edges operation called, there might be that a vertex is connected to the edges is pushed into the priority queue to further be processed again. Then calling process of the priority queue again is O(log V) assuming that all the vertices is pushed to the queue. If all the edges is process too, the total time complexity of operation with edges is O(E.logV). Therefore, the total time complexity of the Dijkstra SSSP algorithm is O(V. log V + E.log V).

Now, time to apply this algorithm to a case, finding a shortest path for users in Google Maps application, from a place to another place with total displacement of 42 km. Along the area from source (initial place) to a destination, there exist shops, fuel station, cross road, etc as vertices with total vertices V = 10.000.000 and the number of streets is the edges and the total edges E = 500.000. Substitute those numbers to the total time complexity and we get O(242.000.751). Assume that a computer can compute 10^8 operation in a second, hence the operation requires ≈ 2,5 seconds to complete the operation. If the displacement between two places is bigger, then the bigger chance that V is getting bigger as well as E. Then this algorithm will be considered as slow enough for a huge-scale tasks.

But the reality is that Google Maps will do this task a lot faster than the time above. From this fact, we can conclude that Google Maps does not use a pure Dijkstra SSSP algorithm, but the programmers of Google Maps implemented "modified" Dijkstra SSSP algorithm, with a lot faster time complexity than pure Dijkstra SSSP algorithm, and the development of this algorithm will further be discussed in the next chapter of this paper.

IV. Development of Dijkstra SSSP Algorithm

Dijkstra Algorithm is quite naïve way of searching the shortest path form an initial node to the other nodes. Considering all of the vertex in the graph until all of the vertex is computed is not well implemented in a large-scale tasks, such as finding a shortest path to a destination in a map, as it takes a long time to find the final answer when the total number of vertices and edges is large enough.

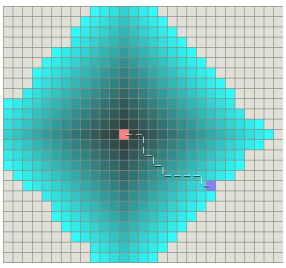

Figure 4: Pure Dijkstra algorithm: total area of pathfinding.
Source: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html accessed in December 9, 2016 at 9.59 a.m.

From this picture, it is obvious that this "naive" algorithm searched all of the possible nodes from the initial node to finally find the destination node (the initial node is pink-marked grid and the destination is marked as

purple). Hence, such a modified Dijkstra SSSP algorithm is needed to do the task well. Modified Dijkstra SSSP algorithm uses heuristic function in order to know whether vertex we are observing is potentially making the way to a certain destination is farther or not. If the unvisited neighbor node is potentially making path to the destination farther, then this node will not be processed. As a consequence, the process of going to the destination node is going to be much faster. But this method has a drawback, it is potentially not going to result the path that has the minimum path from source to destination. Instead of finding the shortest path from the initial node, this modified algorithm focuses on finding the vertex that is closest to the destination, which is hoped to be the closest path as well. One of the algorithm that improvised the Dijkstra Algorithm is A* (A star) algorithm.

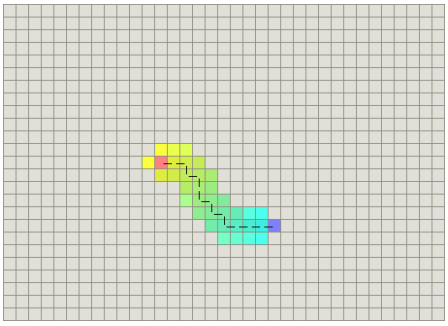

Figure 5: A* algorithm: the way of pathfinding.
Source: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html accessed in December 9, 2016 at 9.59 a.m.

From the picture above, we can see that the A* algorithm uses heuristic functions as a tool and chooses the path that is closest to the goal. With this algorithm, it is guaranteed that will result a less time complexity than the pure Dijkstra SSSP algorithm, but not guaranteed to output the shortest path.

V. Application of Algorithms in Google Maps

Here will be shown how the pure Dijkstra SSSP algorihm if it is implemented in Google Maps, then will be compared if the application uses the heuristic method.works in Google Maps.

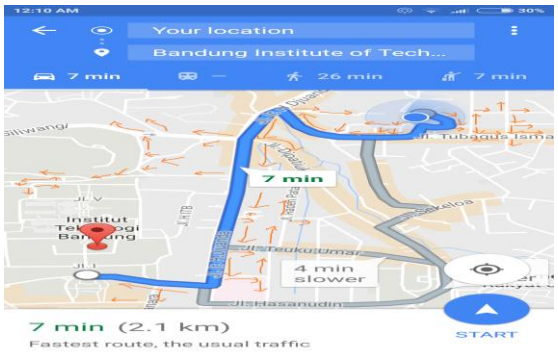

Figure 6: Google Maps path result from Tubagus Ismail V/11 to Bandung Institute of Technology with illustration of pure Dijkstra Algorithm works on it (edited using Paint application on Windows 7).
Screenshot was taken in December 9, 2016 at 12.10 a.m.

In the edited screenshot, orange arrow shows the unvisited neighbor node that is going to be processed through the algorithm. In the picture is shown only part of the process that is observed if using the pure Dijkstra Algorithm, but it shows that there are many nodes that is observed in order of getting to the destination when the destination is only apart 2.1 km from the source. It is surely going to be worse if the distance from the source to the destination is getting bigger as the number of vertices and edges will getting large as well.

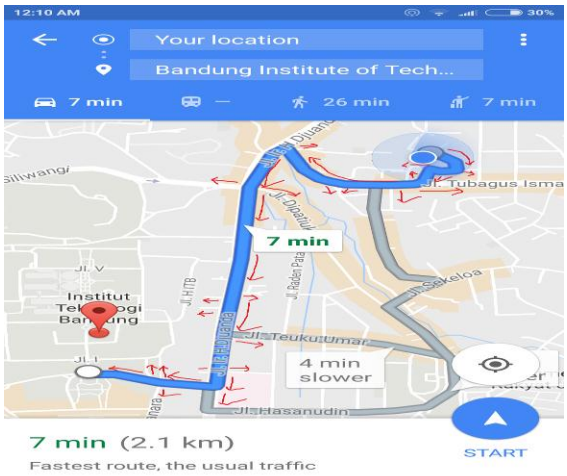

Figure 7: Google Maps result of pathfinding from the source and destination as the figure 6, with the illustration of using heuristic function.
Screenshot was taken in December 9, 2016 at 12.10 a.m.

The picture shows us that using the heuristic function, it will directly go to the unvisited neighbor vertex that is shortest to the destination. Firstly, the unvisited neighbor node is checked, if the node is making the distance to the destination farther, then it will not be processed further. Compared to the figure 6, the red arrow in figure 7 is minimized as heuristic function is implemented in the pathfinding.

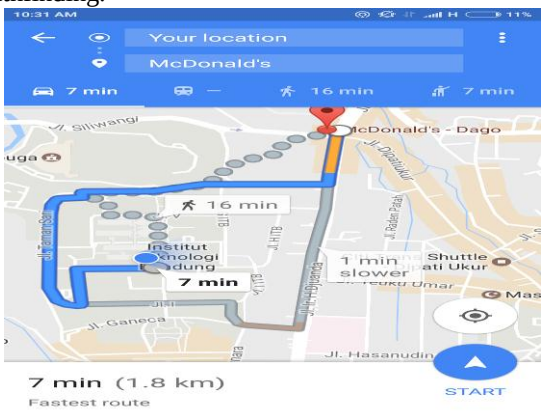

Figure 8: Google Maps path result from Bandung Institute

of Technology to McDonald's Dago.
Screenshot was taken in December 9, 2016 at 10.31 a.m.

The path resulted is shown in the figure 8, when the best way is passing the Tamansari street, then turn right to the Siliwangi street, move straight forward, then turn left and finally reach McDonald's Dago. In fact, the path is different compared to figure 7. It is because the weight of edges (road) that is computed by the app is not only based on the length of the street, but also the street crowd intensity and Google Maps will find the fastest possible route for users. Maybe the street crowd intensity in Ir.H.Juanda is quite high so that it will be taking more time to travel from the initial place to the destination than choosing the path as figured in figure 8 and surely Google Maps will not choose the path in like in figure 7.

Note that in Google Maps, there are colors in the path showing the crowd intensity of each path, where blue is the least crowd intensity, then followed with the higher intensity shown by orange color, then red is showing that the street is extremely crowded and there is traffic jam at the path.

VI. Conclusion

Google Maps, as a pathfinding application, uses Dijkstra SSSP algorithm as a foundation to find a shortest path from an initial place to a certain destination. But using pure Dijkstra SSSP algorithm only is not sufficient to find the path with an extremely short amount of time as it will be slow enough when the distance between source and destination is getting bigger and bigger. Using modified Dijkstra SSSP algorithm, Dijkstra Algorithm using heuristic function is the answer as it is able to reduce the time needed to the destination.

VII. Acknowledgment

Firstly, the author would thanks to God for the strength that He has given to me so that the author has the power to do the task wholeheartedly and also for the passion the author has in computer science. Without His blessings, this paper will not be finished well.

Secondly, the author gives thanks to Luqman Arifin Siswanto as with his opinion on LINE comment after the author randomly posted the title of this paper, it opened the author's mind and searching for another reference to find another better algorithm in pathfinding than pure Dijkstra algorithm.

Lastly, the author gives thanks to all of my friends that accompany the author in doing this task and giving an opinion about my analysis.

References

[1] Munir, Rinaldi, *Diktat Kuliah IF 2120 Matematika Diskrit*. Bandung: 2006, Penerbit Informatika.
[2] Munir, Rinaldi. Slide Perkuliahan IF 2120 Graf (2015).
[3] Pooja Singal, R.S.Chhillar, *Dijkstra Shortest Path Algorithm using Global Positioning System.* International Journal of Computer Application, 2014.
[4] http://techin.oureverydaylife.com/google-maps-work-gps-18373.html, accessed in December 5, 2016 at 9.40 p.m.
[5]]http://everythingcomputerscience.com/algorithms/Dijkstras_Algorithm.html, accessed in December 7, 2016 at 2.02 a.m.
[6] http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html, accessed in December 9, 2016 at 9.59 a.m.

Declaration

I hereby certify that this paper is my own writing, neither a copy nor from another paper, and not an act of plagiarism.

Bandung, December 9, 2016

Dicky Novanto 13515134