CAT 3 REPORT

GALGOTIAS UNIVERSITY

School of Computing Science & Engineering

COURSE NAME: DATA STRUCTURE

COURSE CODE:BCSE2361

SUBMITTED TO: Proff. ABDUL MAZID SIR

SUBMITTED BY:

JANSHI(21SCSE1010892)

SHASHANK SINGH(21SCSE1010310)

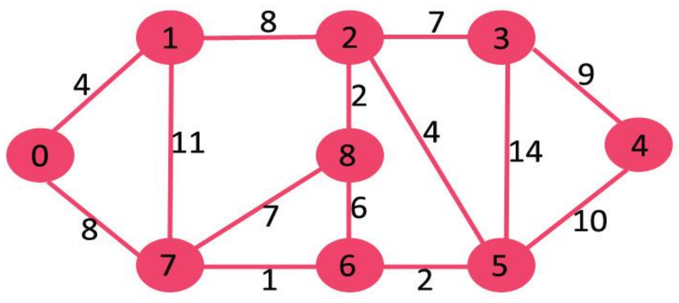
SAKSHI KUMARI(21SCSE1010311)

INTRODUCTION

Single-source shortest route is Dijkstra's algorithm. The single source shortest route problem. It calculates the shortest path from the source to each graph vertex. Single source shortest route problem:

G=V,E is a directed weighted graph with V as the vertices. The source vertex s in V, and for each edge e in E, Edge Cost(e). All graph weights must be positive. Let's discuss directed-weighted network before discussing Dijkstra's algorithm. Directed graphs are ordered pairs G: = (V,E), where V is a collection of vertices or nodes and E is a set of directed edges, arcs, or arrows. Digraphs (directed graphs).

A directed-weighted graph is the same as a directed graph, except that each edge of the graph has a weight associated with it.



Dijkstra’s –A Greedy Algorithm

Greedy algorithms are computer programmes that use methods of problem solving that are based on activities to decide whether or not there is a better long term approach. These programmes decide whether or not there is a better long term approach based on the activities that are being performed. Among these efforts is the investigation of whether or if there is a superior approach for the long run. Dijkstra's method takes a greedy approach to the problem of locating the one source that is connected to all other sources by the shortest path. This problem can be solved by finding the source that is connected to all other sources by the shortest path. It always chooses, from among the vertices that have not been selected yet, the vertex that is geographically closest to the source, and then, once it has made this selection, it declares the distance between the two to be the genuine shortest distance.

DESCRIPTION OF THE ALGORITHM

It is absolutely necessary to have a solid understanding of the algorithm's operation before delving into the mechanics of the pseudocode for the algorithm. Dijkstra's algorithm functions by first resolving a subproblem known as "k," which computes the shortest path from the source to vertices that are among the k vertices that are located the closest to the source. This is done so that the algorithm can move on to the main task of finding the path with the fewest number of hops. This step is taken so that the algorithm may proceed to the more important task of determining the path with the least distance. In order for Dijkstra's technique to successfully operate, the graph in question needs to be directed and weighted, and the edges can't have a value that is less than zero. Additional prerequisites for the graph are as follows: If the values of the edges are negative, it is not possible to find the route that has the least distance since there is no conceivable way to do so.

Following the completion of the kth iteration, there will be a collection of vertices that is referred to as the Frontier. This collection will be made up of the vertices that are located in the places that are the farthest away from the point where the data was initially gathered. Calculations will be done on vertices that are located outside of the border region, and the coordinates of these vertices will be added to the set that represents the New Frontier. The variable that is designated by the letter w will always have the value of sDist[w] assigned to it. This value will be the shortest distance that can be calculated. It offers a rough estimate of the possible distance that may be travelled from east to west.

When using Dijkstra's approach, the vertices of the New Frontier are stored in a priority-min queue. This allows for the determination of the vertex that is the next nearest to the current position.

In order for the method to work as designed, it is necessary to record the shortest distance separating vertex v from the source in an array with the name sDist. This is done to ensure that the operation achieves the desired results. The distance that separates the source and itself is precisely 0 units, making it the distance with the smallest possible difference between the two points.

The value of sDist has been changed to infinity for each and every one of the other vertices in order to denote that the processing of those vertices has not yet been completed. This is done in order to make the graph simpler and more straightforward to comprehend. After the algorithm has finished processing the vertices, sDist will be updated to include the value that indicates the smallest possible distance that exists between vertex w and s. This value will be the shortest distance that exists. Because both the Frontier and the New Frontier sets are kept at the most recent versions possible, the algorithm may be processed as rapidly and efficiently as possible.

Frontier owns k vertices that are located in the closest proximity to the source, and it will have already computed the shortest distances between these vertices, for paths that are restricted to a maximum of k vertices in total. This is because Frontier has already determined the shortest distances between these vertices. The collection of vertices that can be discovered in regions other than Frontier is referred to as the New Frontier collection.

PSEUDO-CODE OF THE ALGORITHM

The pseudo-code that is presented below offers a concise explanation of how Dijkstra's algorithm accomplishes its tasks.

Procedure (Vertex 1 is the source) Dijsktra (V: a collection of vertices)

Adjectives [1–n] referring to adjacency lists;

EdgeCost(u, w) is a function that calculates the edge's cost.

Variable: the path costs sDist[1...n] from the source (vertex 1); the value of sDist[j] will be equal to the length of the shortest path to vertex j.

Begin: \sInitialize

Create a virtual set called Frontier to hold I in cases where the problem with sDist[i] has already been entirely handled.

Make sure the Priority Queue in New Frontier is empty;

sDist[1]=0; "The distance to the source is zero" for all of the vertices that are in V – "1" does "no edges have been investigated yet"

sDist[w] as a conclusion for;

Vertices should be placed in New Frontier in the order of their respective priority. sDist[w]; \send Initialize; repeat vDeleteMinNew Frontier; sDist[v] is already accurate; v is the new closest.

for all of the people that live nearby w in Adj[v] do if sDist[w]> sDist[v] +EdgeCost(v,w), and if so, then sDist[w] sDist[v] +EdgeCost (v,w)

rewrite section w in New Frontier to reflect the new priority. sDist[w]}

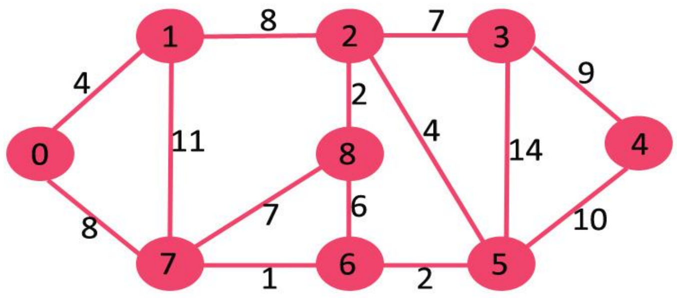
endif endfor until New Frontier is empty end

Dijkstra; This example shows how the algorithm works Best-First-Breadth-First-Search. The search that is carried out is a Breadth-First one because New Frontier is made up of vertices that can be tried next and these vertices are only one edge away from the vertices that have already been explored. This causes it to be a Best-First search because the best vertex in New Frontier is chosen to be processed next.

EXAMPLE

Find the shortest path from source to all nodes in a network to understand Dijkstra's Algorithm.

Consider the graph below with src = 0.



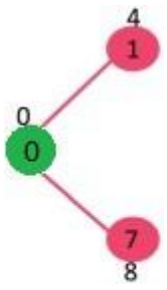
SptSet is initially empty and vertex distances are 0, INF, INF, INF, INF, INF, INF.

STEP 1:

Now choose the closest vertex. Include vertex 0 in sptSet. So sptSet=0. Update sptSet's vertex distances after adding 0.

0 has 7 adjacent vertices. 1 and 7 are now 4 and 8.

The following subgraph shows vertices with finite distances. SPT's green vertices.

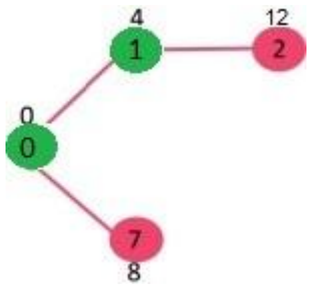


STEP 2:

Choose the vertex that has the smallest distance value and is not already accounted for in the SPT. A selection is made, and vertex 1 is added to the sptSet.

Therefore, sptSet will now be written as "0, 1." Make sure that the values of the distance between adjacent vertices are all set to 1.

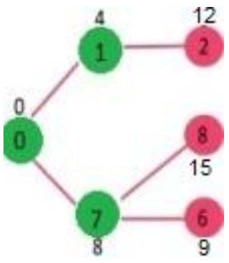
The value of the distance between vertices 2 and 3 is now 12.



STEP 3:

Choose the vertex that has the smallest distance value and is not already accounted for in the SPT (not in sptSET). The vertex number 7 is selected. Therefore, sptSet will now be written as "0, 1, 7."

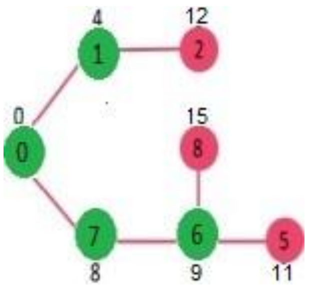
Keep the values of the distance between adjacent vertices of 7 up to date. The value of the distance between vertices 6 and 8 can be considered to be finite.



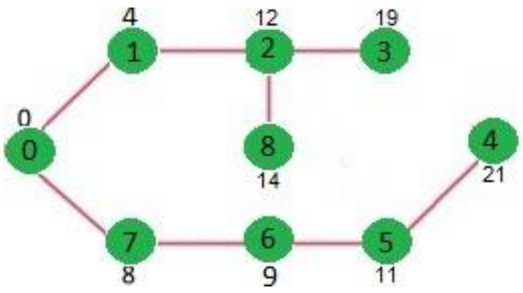
STEP 4:

Choose the vertex that has the smallest distance value and is not already accounted for in the SPT (not in sptSET). The vertex number 6 is selected. Therefore, sptSet will now be written as 0, 1, 7, 6.

Keep the values of the distance between neighbouring vertices of 6 up to date. The value of the distance between vertices 5 and 8 has been adjusted.



The shortest which we got:



APPLICATIONS AND EFFICIENCY

APPLICATION:

The Dijkstra algorithm is utilised by traffic information systems in order to trace the origin and destination of traffic originating from and terminating at a specific origin and destination.

• OSPF stands for Open Shortest Path First and is a routing protocol used on the internet. Within each of the different regions that comprise the hierarchy, it implements a link-state. The computation is based on Dijkstra's algorithm, which determines the shortest path tree within each section of the network. This algorithm is used to calculate the shortest path.

EFFICIENCY:

The complexity as well as the efficiency may both be stated using the Big-O Notation. The Big-O notation provides an additional way to discuss the influence that input has on the execution time of an algorithm. It provides a maximum estimate of how long it will take.

The efficiency of the Dijkstra algorithm shifts depending on the values of |V|=n DeleteMins and |E| for the updates to the priority queues that were implemented.

The complexity would be O(| E | + | V | log | V |) if a Fibonacci heap was employed, which is the best bound possible. The time complexity of the DeleteMins operation is O(log|V|).

If an ordinary binary heap is utilized, then the complexity is denoted by the notation O(| E |log |E|), where the notation| E | log |E| term originates from| E|updates for the ordinary heap.

The complexity is denoted as O(|E|+|V| 2) if the set that is being utilised is a priority queue. The O(V| 2) term is generated by doing V scans on the unordered set New Frontier in order to locate the vertex that has the smallest sDist value.

REFRENCES

http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/Dij kstraApplet.html

http://tide4javascript.com/?s=Dijkstra

http://www.unf.edu/~wkloster/foundations/DijkstraApplet/DijkstraAppl et.htm