Advanced study of Shortest Route Problem and its applications- bellman – ford algorithm

[1]Ramesh lempepatil, [2]Vishwas V Rudraswamymath

Department Mathematics
Sharnbasva University Kalaburagi, India.

Abstract- Numerous shortest path graph analysis algorithms fall within the single source, single destination, or all pairs' categories. When there is a single source, one such approach that fits into the first category is Dijkstra's algorithm, which can be used to determine which node has the lowest cost. Negative weight cannot be used, though. In some scenarios, it can merely fail to deliver the desired consequences. However, we must identify the shortest route between the networks in this project. For this, a number of algorithms exist, including Bellman-Ford, A*, and Floyd-Warshall. It is a single source shortest path issue in Bellman-Ford, similar to Dijkstra's, However, it has edges with negative weight. In the Floyd-Warshall algorithm, each node has the ability to function as a source and can be used to find the shortest path between source and destination nodes, Floyd-Warshall will accurately assert that there is no least weight path in the event of a negative cycle since the negative weight is unbounded. To put it briefly, Floyd-Warshall is a suitable method for our system and will be used to generate the best route for every node pairing.

CHAPTER-1
SHORTEST PATH ALGORITHM
INTRODUCTION:

Networks can develop in a variety of contexts and ways, including in the ubiquitous Our daily lives are impacted by the transportation, power, and communication networks. The representations of networks frequently used to solve issues in a variety of fields, including resource management, project planning, facility location, transportation, production, and distribution of oil, among others. It offers strong conceptual and visual support that is utilized to illustrate the interactions between the parts of systems in almost every area of scientific, social, and economic effort. The extraordinarily quick advancement in both the technique and implementation of network optimization models over the past few years has been one of the most fascinating breakthroughs in operation research. Data structure and effective data manipulation have greatly benefited from a number of algorithmic innovations. due to algorithms and Since software is now easily accessible, many formerly completely impractical routines can now be resolved. The model takes into account everya component of the company, assisting management in planning and choosing varying degrees at variousphases to the industry, such asknowing what to pay and what to charge. Because it aids in determining and observing the flow of commodities from the industry to their destination, a network representation is crucial to an industry.

Graph theory has emerged as a significant tool for representing and analyzing a range of real-world issues. A graph in this sense is composed of vertices.known as nodes and the connecting lines known as edges. In this project, the shortest path algorithm is being used to find asolution for our system. In general, Any shortest path algorithm's categories can be split into three groups: single source, single destination, and all pairings. The shortest route in a directed graph from each vertex to The single source leads to a single destination.approach. By flipping the arcs, it can also be understood as a single source of issues. The final group, all pairs of vertices, findevery pair of vertices' shortest path, if there are negative edges present, the one source problem can be handled using algorithms like Dijkstra's and Bellman-Ford. The goal of every pair method, including Floyd-Warshall, is to identify the shortest pathamong each pair of vertices. On the other hand, Floyd-Warshall is more helpful when dealing with thick graphs, such as actual road networks.

SHORTEST PATH PROBLEM:

The goal of the shortest path problem in graph theory is to find a path between two vertices (nodes) of a graph where the weight of each of its individual edges is reduced.

SHORTEST PATH ALGORITHM:

The group of algorithms referred to as "shortest path algorithms"is created to address the shortest path issue. Most people have some intuitive acquaintance with the shortest path problem at this point. In computer science, the shortest path issue can take on a variety of different shapes, necessitating the use of multiple techniques to resolve it.

In order to be straightforward and universal, shortest path algorithms often operate on an input graph G. A several vertices V and the connecting edges E make up the graph. The graph is referred to asa graph that is weighted if the edges are weighted. The graph is referred to as undirected when these edges are bidirectional. Even cycles can appear in a graph occasionally. Each This distinction is what allows one algorithm to perform better than another on a certain type of graph.

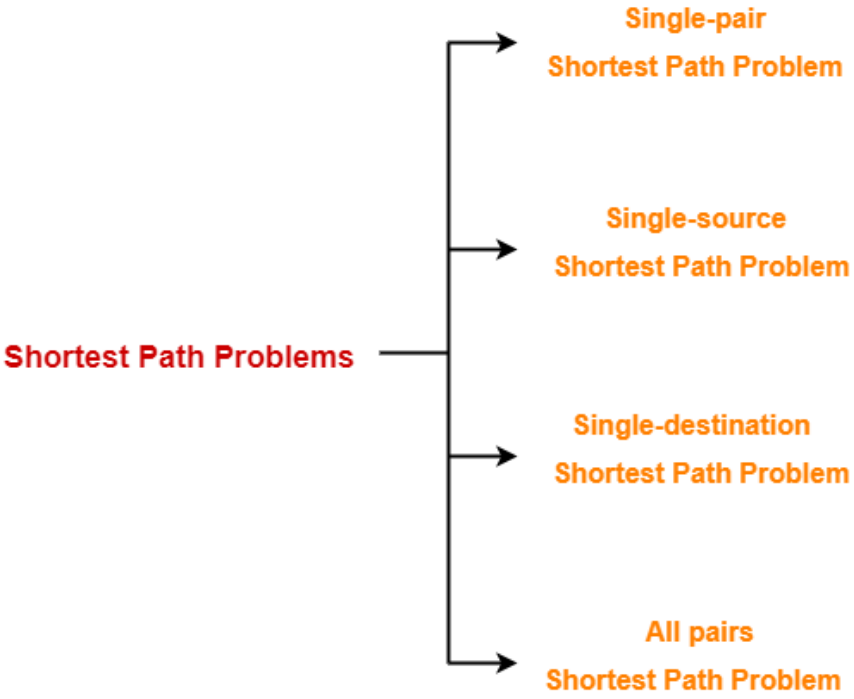
The shortest path algorithm has several applications. As was already established, mapping programmer like Google or Apple Maps use shortest path algorithms. The road network, operations and logistics research, and computer networks like the internet all depend on the use of shortest path algorithms.

Any software that directs your route selection use some sort of shortest path algorithm. If you give Google Maps a beginning point and an ending point, it will determine the shortest route.

SHORTEST PATH PROBLEM TYPES:

The following are some examples of shortest path issues:

1. Problem of the single-pair shortest path
2. Problem of the Single-Source Shortest Path
3.Single-Destination shortest path problem
4. The shortest path problem for all pairs



Problem of the single-pair shortest path

1. This shortest path issue involves calculating the distance between a pair of supplied vertices.
2. The well-known A*search technique is utilized to resolve the single pair the shortest path issue.

Problem of the Single-Source Shortest Path

1. This shortest path problem determines the shortest path from a specific source vertex to each of the remaining vertices.
2. Well-known techniques for resolving The Dijkstra algorithm and the Bellman Ford method are two solutions to the Single-Source Shortest Path Problem.

Single-Destination shortest path problem:

1. Theshortest route between each vertex and a single destination the single-destination shortest path problem involves the calculation of vertices.
2. The problem can be simplified to a single source shortest path problem by flipping the direction of each graph edge.
3. The shortest route problem with a single destination is resolved by the well-known Dijkstra's Algorithm.

The shortest path problem for all pairs:

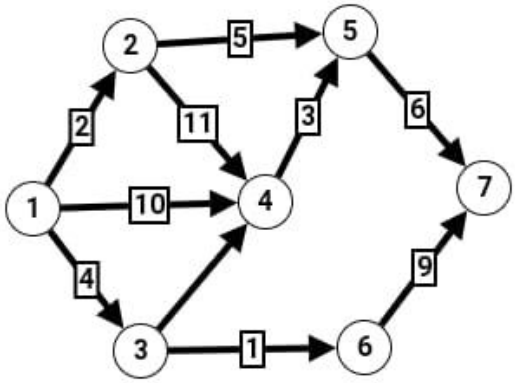
1. This shortest path problem determines the shortest path between each pair of vertices.
2. Floyd-Warshall and Johnson's algorithms are two popular strategies for resolving the All-Pairs Shortest Path Problem.

Steps for Shortest path problem:

1. Assign variables to network nodes whose labels correspond to the nodes shortest distances from a given origin.
2. Assign origin as node-1, label it permanently as u_1=0, and assign'm' to temporarily control the other nodes.
3. We compute for temporary label as $u_j = (u_j, u_i + d_{ij})$ where node-j is momentarily labelled and distance d_{ij} smaller than m.
4. Node-1 is permanently labelled and other nodes are briefly marked as m from node-1. Locate the smallest component of the temporary label, such asu_i, and node-i will now be permanently labelled withu_i.
5. If every node is permanently nodded, or if every node is permanently labelled "fixed shortest path," then step 3 should be taken.

Example on Shortest path problem

Find the shortest route between each of the other nodes and node 1



Solution: Assign u_1= 0. Take it as permanent node. From node-1 it can reach node-3, node-2, and node-4. $u_2 = u_3 = u_4$

Node (2) =$u_2 = \min(u_2, u_1 + d_{12})$
= min (m, 0+2)
u_2 = 2

Node (3) =$u_3 = \min(u_3, u_1 + d_{13})$
= min (m, 0+4)
u_3 = 4

Node (4) =$u_4 = \min(u_4, u_1 + d_{14})$
= min (m, 0+10)
u_4= 10

The smallest labeled is u_2 = 2

u_2 Is permanently labeled. From u_2 we are reaching to u_4 and u_5

Node (4) =$u_4 = \min(u_4, u_2 + d_{24})$
= min (m, 2+11)
u_4= 13

Node (5) =$u_5 = \min(u_5, u_2 + d_{25})$
= min (m, 2+5)
u_5= 7

The smallest label is 7 i.e., u_5= 7 is permanently labeled.

From node-5 we can reach to u_7 only,

Node (7) =$u_7 = \min(u_7, u_5 + d_{57})$
= min (m, 7+6)
u_7= 13

Therefore, the smallest path is

$1 \rightarrow 2 \rightarrow 5 \rightarrow 7$

$\therefore$ The cost is $2 + 5 + 6 = 13$

CHAPTER-2

ALGORITHM ANALYSIS

Dijkstra's Algorithm:

Dijkstra's algorithm is one of the most popular methods for finding the shortest path between any two vertices on a graph or for solving a variety of single-source shortest path issues with non-negative edge weight in graphs.

Dr. Edsger W. Dijkstra, a talented Dutch computer scientist and software developer, developed and published this technique.

His innovative technique was described in a three-page article he wrote in 1959 titled "A comment on two problems in connection with graph"

Dijkstra's Algorithm's fundamentals

- The source node you choose is where Dijkstra's Algorithm begins, andexamines the graph to identify the node's shortest route to each other node in the network.
- The algorithm modifies the reported shortest distance between each node and the source node if it finds a shorter path.
- Once the algorithm has identified the shortest path between a node and another node, the node is added to the path and marked as "visited"
- The process will continue until all of the graph's nodes have been added to the path. Now we have a path that in this way connects the source node to every other node.

Only positive weight graphs can be used with Dijkstra's algorithm. This is because in order to identify the shortest path, the weight of the edges must be applied during the procedure.

The algorithm won't function properly if the network contains a negative weight. The current route to a node is designated as the quickest route to that node once it has been registered as "visited." And if the total weight is reduced after this step, negative weights can change this.

Define $d_{ij} > 0$, and let u_i be the shortest distance between source node 1 and node i. The algorithm specifies the label for a node j that succeeds it immediately as the length of r (i, j).i.e, $[u_j, i] = [u_i + d_{ij}, j], d_{ij} \geq 0$

The initial node's label reads [0, -], designating it as the predecessor. There are two sorts of labels in the Dijkstra algorithm: 1. Temporary 2. Consistent

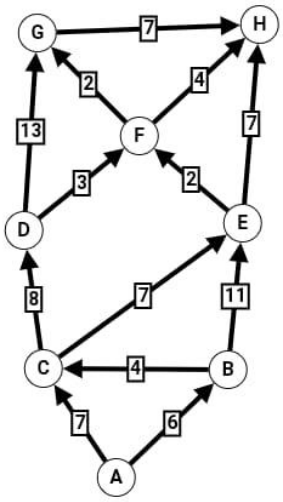
1. If the shortest path to the node can be found otherwise, a temporary label at a node is changed to a permanent one.
2. Permanently affix the label [0 -] from set i=1 to source node 1.

Basic Actions

1. calculate the interim labels $[u_i+d_{ij}, j]$ for each node j with $d_{ij} > 0$ provided j is not permanently labeled. If node j already has an existing temporary label $[u_j,k]$ through another node k and if $[u_i+d_{ij}, j] < u_i$ replace $[u_j,k]$ with $[u_i+d_{ij}, j]$
2. If every node has a permanent label, the operation should be stopped; otherwise, choose the temporary label $[u_r,s]$ with the shortest distance from all the other temporary labels, set i=r, and go back to step 1.

Example:

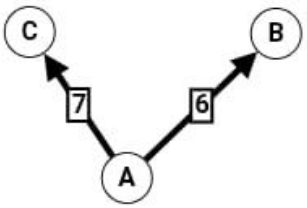
The network provides a path that is permeable as well as their lines in kilometers between city 1 (node 1) and seven other cities (node2, node3, node4, node5, node6, node7, node8) The shortest path between city1 and each of the other five cities should be



found.

Solution;Put [0 -] as the label's permanent form.

Iteration 1:

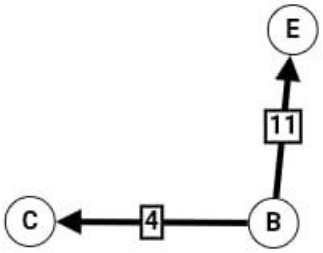


The list of labelled nodes becomes shorter since Node A (the last permanent labelled node) can be reached from Nodes B and C. (temporary and permanent).

NODES	LABEL	STATUS
A	[0 A]	Permanent
B	[0+6, A]	Permanent
C	[0+7, A]	Temporary

Because node B is closer to the two temporary labels [6, A] and [7, A] (u_B=6), its status has been changed from temporary to permanent.

Iteration 2:



Since Node B permanent labelled node can be reached from Nodes C and E, the list of labelled nodes becomes (temporary and permanent).

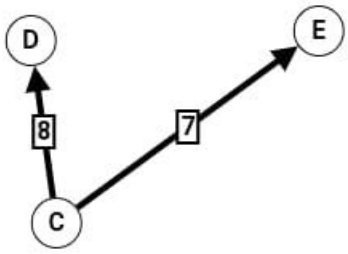
NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Temporary
C	[6+4, B]	Temporary
E	[6+11, B]	Temporary

Node C is a temporary label [7, A] received in iteration 1 remains the same since node 3 retains another label [10, B] in iteration2 and the label [7, A] has the lowest distance.

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[17, B]	Temporary

Since node C offers the smallest distance, u_C=7, it has been made permanent.

Iteration 3:



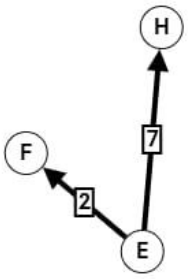
Node C permanently lebeled and it extended to Node D and Node E

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[17, B]	Temporary
D	[7+8,C]	Temporary
E	[7+7,C]	Temporary

Node E is temporary labeled from iteration2 is thus [14,C] obtained from iteration 3. Thus Node E yields shortest distance u_E = 14 thus we change node E to permanently labeled. The list of labeled becomes;

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15,C]	Temporary

Iteration 4:



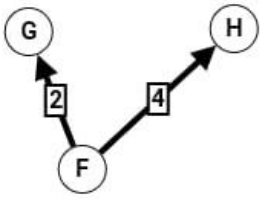
Node E connected to Node F and node H. Thus, the list of labeled node becomes;

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Temporary
F	[14+2,E]	Temporary
H	[14+7,E]	Temporary

Node F submit the shortest distance from node E, thus node F u_F=16 becomes permanent. The list of label becomes;

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Temporary
F	[16,E]	Permanent
H	[21,E]	Temporary

Iteration 5:



Node F is connected to node G and node H. The list of labeled node becomes;

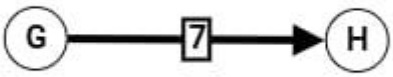
NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Temporary
F	[16,E]	Permanent
H	[21,E]	Temporary
G	[16+2, F]	Temporary
H	[16+4,F]	Temporary

Node H is temporary labeled [20, F] obtained from iteration 5, [21, E] is obtained in iteration 4 indicates the shortest route through nodeE. The shortest distance is node G (u_H=20) labeled [20,F], thus we have

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent

D	[15, C]	Temporary
F	[16,E]	Permanent
H	[20,F]	Temporary
G	[18, F]	Permanent

Iteration 6: Node H is connected to Node G



NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Temporary
F	[16,E]	Permanent
H	[20,F]	Temporary
G	[18, F]	Permanent
H	[18+7,G]	Temporary

In iteration 5 Node H is temporarily labeled [20,F], in iteration 6 it is [25,G], then the smallest one is [20,F]

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Temporary
F	[16,E]	Permanent
H	[20,F]	Permanent
G	[18, F]	Permanent

Iteration 7:

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Temporary
F	[16,E]	Permanent
H	[20,F]	Permanent
G	[18, F]	Permanent

We make node D (u_D=15) status permanent because node D can only be accessed by node C.

NODE	LABEL	STATUS
A	[0, A]	Permanent
B	[6, A]	Permanent
C	[7, A]	Permanent
E	[14, C]	Permanent
D	[15, C]	Permanent
F	[16,E]	Permanent
H	[20,F]	Permanent
G	[18, F]	Permanent

All of the lists now have a permanent status.
As a result, the answer to the question of what is the shortest distance between node A and any other node in the network is

NODES	ROUTES	DISTANCE

A	A	0
B	A-B	6
C	A-C	7
D	A-C-D	15
E	A-C-E	14
F	A-C-E-F	16
G	A-C-E-F-G	18
H	A-C-E-H	21

Applications of Dijkstra's Algorithm

Dijkstra's algorithm has several practical applications, some of which are listed below:

1. **Google Maps' digital mapping services:** G-Maps has been used numerous times to calculate the distance between two cities or between your current location and the closest desirable destination. The Shortest Path Algorithm is used because there are many ways to get from one place to another, yet it needs to show the shortest distance. The shortest distance between any two places on the path is found using Dijkstra's Algorithm. This method can be used to find the shortest routes inside India, between any two cities or locations, as well as between any two cities or locations. Imagine India as a graph, with each city or location representing a vertex, and the distance between each vertex and each location as an edge.
2. **Social networking applications:** Many programmes include a list of friends that a given user might know, as you may have observed. How successfully do you think many social media companies will integrate this feature given that the system has more than a billion users? The traditional Dijkstra algorithm can be used to find the user-to-user shortest path, as indicated by handshakes or connections between them. When the social networking graph is tiny, various factors are used with the traditional Dijkstra's algorithm to determine the shortest pathways. The employment of alternative sophisticated algorithms is necessary since the regular approach becomes increasingly slower as the graph gets bigger.
3. **Telephone Network**:Every line in a telephone network has a bandwidth, ab, as we are all aware. Depending on its bandwidth, a transmission line can support frequencies up to a certain limit. Typically, the signal is weaker along a certain line if the frequency of the signal is higher along that line. Bandwidth is a unit of measurement for the volume of data that can be sent through a line. Switching stations would be the nodes, transmission lines would be the edges, and the weight of the edges would be the value "b" if we were to visualize a city as a graph. As you can see, the Dijkstra algorithm can be used to resolve it. It can be categorized as a shortest distance problem.
4. **Find IP routing Open Shortest Path First (OSPF)** is a link-state routing protocol that chooses the best path between the source and destination routers using its proprietary Shortest Path First algorithm. Dijkstra's technique is extensively used in routing protocols, which routers utilize to update their forwarding tables. The method presents the cheapest path away from the starting router to the other routers in the network.
5. **Flighting Agenda:** Consider the situation when someone needs software to generate a flight itinerary for customers. The agent has access to a database of all airports and flights. Along with the flight number, origin airport, and destination, the flights also offer the departure and arrival timings. Given the airport of origin and the start time, the agent wishes to know the earliest arrival time at the destination. There, this algorithm is employed.
6. **Choose a file server**: On a LAN, a file server can be chosen using Dijkstra's algorithm (local area network). Keep in mind that transferring files across computers takes an interminable length of time. The objective is to use Dijkstra's algorithm to reduce the shortest path between the networks, producing the fewest possible "hops" from the file server to each other computer on the network.
7. **Robotic Path**: It is now possible to use robots and drones, some of which are automated and others of which are manual. The robot or drone moves in the requested direction by choosing the shortest path when the source and destination are known, continuing to deliver the cargo in the quickest period of time. Automated drones and robots that are employed for jobs or to deliver products to certain areas are equipped with this algorithm module.

BELLMAN – FORD ALGORITHM:

In a weighted directed network, the shortest path is found using the Bellman-Ford algorithm, a single source method.The shortest path between one source and several destinations is what we mean when we say that a single source algorithm exists. Contrary to Dijkstra's, it takes into account even negative edge weights. In this approach, a low value path can be formed if. All edges are simply relaxed by this algorithm until |v|-1 is reached. The number of vertices with accurately calculated distances increases with repetitions, and eventually all vertices will have accurate distances. This is one of the explanations for the wider class of inputs to which this method is used. Additionally, it improves system performance by allowing traffic to be dividedacross multiple routes. This algorithm struggles to scale. Network topology can be altered, however due to the slow propagation of information from node to node, the modifications are not immediately apparent. Additionally, if a node fails and becomes inaccessible to some other nodes, those nodes may spend all of eternity increasing their estimations in order to reach the failed node. These problems are known as the "count to infinity" problems.

Find the shortest routes between the given graph's source vertex (src) and each of its vertices.

Negative weight edges could be present in the graph

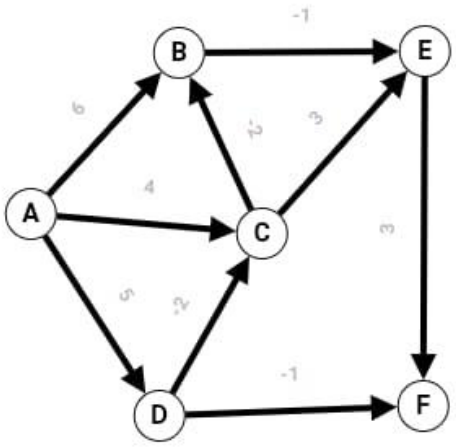
ALGORITHM:

Graph input with src as the source vertex

Shortest distance from src to all vertices is the output. The shortest distance is not determined and a negative weight cycle is given if there is one.

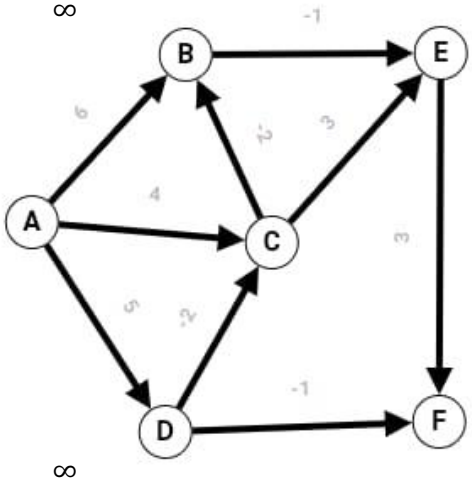
1. In this stage, the distance to the source is set to 0 and the distances to all other vertices are initialized as infinite. These values will be kept in the array dist [] of size v.
2. In order to determine the shortest distances, multiply the following by |v|-1 Follow these steps for each edge u-v
   If dist. [v] >dist. [u] + weight of u-v
   dist. [v] = dist. [u] + the weight of the u-v
3. This stage informs you if the graph has a negative weight cycle.
   Follow these steps for each edge u-v
   There is a negative weight cycle in the graph if dist. [v] >dist. [u] + weight of edge u-v.

Example:



Take A as the source vertex, which we will set to 0. Set all source distances to their initial values. Since there are 6 total vertices in the graph, all edges must be processed 4 times.

∞ ∞



0 ∞ ∞

A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Process all edges as follows: (A, B), (A, C), (A, D), (B, E), (C, B), (C, E), (D, C), (D, F) (E, F). When all edges are first processed, the following distances are obtained. Using the equationdist. [v] =dist. [u] + weight of edge u-v

Taking from (A, B)

dist[B] = dist[A] + weight of edge A-B

Taking from (A, D)

dist[D] = dist[A] + weight of edge A-D

Taking from (C, E)

dist[E] = dist[C] + weight of edge C-E

Taking from (D, F)

Taking from (C, B)

dist[B] = dist[C] + weight of edge C-B

d(B)= 5-1=4

Taking from (A, C)

dist[C] = dist[A] + weight of edge A-C

Taking from (B, E)

Taking from (D, C)

dist[C] = dist[D] + weight of edge D-C

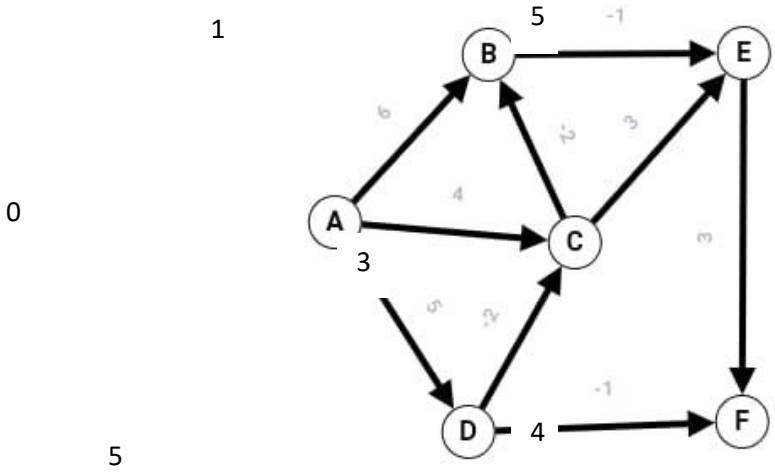
Taking from (E, F)

dist[F] = dist[E] + weight of edge E-F

d(F)= 5+3=8

Taking the minimum distance from the above distance

A	B	C	D	E	F
0	1	3	5	5	4



Process all edges as follows: (A, B), (A, C), (A, D), (B, E), (C, B), (C, E), (D, C), (D, F) (E, F). When all edges are second processed, the following distances are obtained. Using the equation

Taking from (A, B)

dist[B] = dist[A] + weight of edge A-B

Taking from (A, D)

dist[D] = dist[A] + weight of edge A-D

Taking from (C, E)

dist[E] = dist[C] + weight of edge C-E

d(E)= 3+3= 6

Taking from (A, C)

dist[C] = dist[A] + weight of edge A-C

Taking from (B, E)

dist[E] = dist[B] + weight of edge B-E

Taking from (D, C)

Taking from (E, F)

dist[F] = dist[E] + weight of edge E-F

d(F)= 0+3=3

Taking from (C, B)

dist[B] = dist[C] + weight of edge C-B

d(B)= 3-2=1

A	B	C	D	E	F
0	1	3	5	0	3

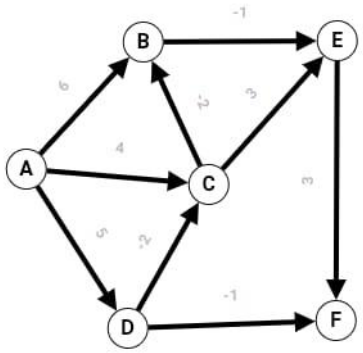
1

0

0

3

5



3

Taking from (A, B)

dist[B] = dist[A] + weight of edge A-B

d(B)= 0+6= 6

Taking from (A, C)

dist[C] = dist[A] + weight of edge A-C

Taking from (A, D)

Taking from (B, E)

dist[E] = dist[B] + weight of edge B-E

Taking from (C, E)

dist[E] = dist[C] + weight of edge C-E

Taking from (D, C)

dist[C] = dist[D] + weight of edge D-C

Taking from (D, F)

dist[F] = dist[D] + weight of edge D-F

Taking from (E, F)

dist[F] = dist[E] + weight of edge E-F

Taking from (C, B)

d(F)= 0+3=3

dist[B] = dist[C] + weight of edge C-B

d(B)= 3-2=1

Taking minimum distance from the above distance.

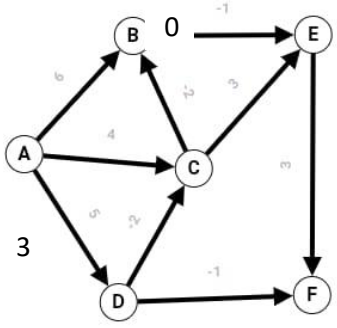
A	B	C	D	E	F
0	1	3	5	0	3

1

0

0

3



After third iteration, the second and third iteration are same the no need to perform all iteration. The values remain same.

3

5

Vertex	Distance from Source
A	0
B	1
C	3
D	5
E	0
F	3

Applications of Bellman-Ford Algorithm:

• It is utilized by distance-vector routing techniques like Routing information protocols (RIPs).
• The maximum weighted matching problem employs it.
• It is applied to the shortest path issue for all couples.
• Calculating the least amount of heat gain or loss during chemical processes.
• Determining the most effective currency conversion technique.

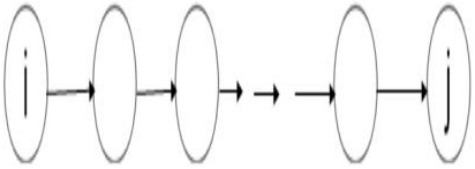
FLOYD'S ALGORITHM:

The Floyd-Warshall algorithm is also known as Floyd's and Roy-Warshall algorithms.The shortest path in a weighted graph can be found using this graph analysis approach, which is also used to identify the transitive closure of the relation R. Edge weights on a weighted graph might be either positive or negative (but with no negative cycle). A single execution of the procedure calculates the total length of all the shortest paths connecting every single pair of vertices in the graph; as a result, it does not return information about the actual paths. The Floyd-Warshall algorithm will be applied to locate every pair of vertices in the graph. This approach uses adjacency matrices rather than adjacency lists and is competitive for dense graphs. Consider the following example of TSP provided by W: Display the directed, edge-weighted adjacency matrix of the graph.

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix}$$

w_{ij} is the edge's weight (i,j), or infinite if there isn't an edge at all.

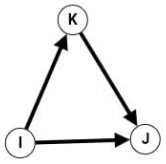
Give back a matrix D with each entry being d_{ij} is d(i,j). might potentially return a predecessor matrix, P, where each item, p_{ij}, represents j's shortest-path predecessor on the left. Think about a path's intermediate vertices;



Assume for the moment that we are aware of the shortest path's length, which only has intermediate vertex numbers 1, 2, 3, k-1. Floyd's algorithm, which finds the shortest path between any two network nodes, is more versatile than Dijkstra's approach. The approach yields the finite distance d ij from node I to node j and represents a n node network as a square matrix. If I and j are directly connected and ∞otherwise.

It is shortest to go through k when there are three nodes, I j, and k, with connection lengths listed on.

d_{ik} d_{kj}



d_{ij}

If $d_{ik} + d_{kj} = d_{ij}$

Applying this triple operation exchange on the distance is done using the following steps:matrix in order to replace the direct route from i to j with the indirect route I to k, k to j,

(i → k → j)

Step 1: Set k=1 and define D_0 as the shortest distance matrix and node square matrix, respectively.

$$D_0 = \begin{array}{c|cccccccc|} & 1 & 2 & 3 & \ldots\ldots & j & \ldots\ldots & n \\ \hline 1 & - & d_{12} & d_{13} & \ldots\ldots & d_{1j} & \ldots\ldots & d_{1n} \\ 2 & d_{21} & - & d_{23} & \ldots\ldots & d_{2j} & \ldots\ldots & d_{2n} \\ 3 & d_{31} & d_{32} & - & \ldots\ldots & d_{3j} & \ldots\ldots & d_{3n} \\ : & : & : & & & & & \\ : & : & : & & & & & \\ i & d_{i1} & d_{i2} & d_{i3} & \ldots\ldots & & & d_{in} \\ : & : & : & : & & & & \\ : & : & : & : & & & & \\ n & d_{n1} & d_{n2} & d_{n3} & \ldots\ldots & & & d_{nn} \end{array}$$

$$\begin{array}{c|cccccc|} & 1 & 2 & \ldots.j & \ldots\ldots & n & \\ \hline 1 & - & 2 & \ldots\ldots & j & \ldots\ldots & n \\ 2 & 1 & - & \ldots\ldots & j & \ldots\ldots & n \\ 3 & \vdots & : & - & : & : & : \\ : & : & : & : & : & : & : \\ : & 1 & 2 & \ldots\ldots & j & \ldots\ldots & - \end{array}$$

Step 2 : Apply the triple operation to each element d ij in D (k-1) for all I & j such that. Define row k and column k as pivot rows and pivot columns.

$d_{ik} + d_{jk} < d_{ij} \quad (i \neq k, j \neq k)\ \&(i \neq j)$

Pivot	Row	…..	d_{ij}	d_{ik}	d_{iq}
	Row	…..	d_{kj}	:	d_{kq}
	Row	…..	d_{pj}	d_{pk}	d_{pq}

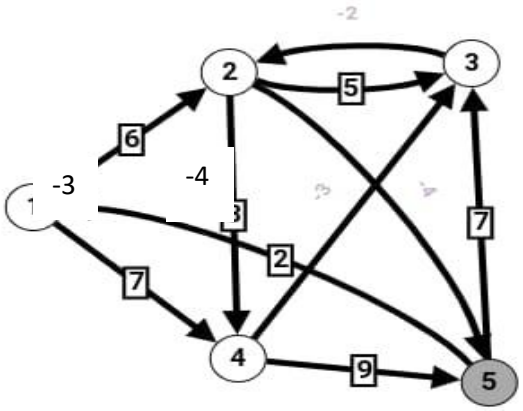
Step 3 : S (k-1) becomes s k by substituting s ij with k. By expressing D (k-1) here, row k and column k determined by the current pivot row and columnD_{ij}=min(D_{ij}, D_{iK}+D_{jK}), step 2 of the method can be described. Set k = k+1, If k = m+1 then halt, else repeated step 2.

If some of the pivot row & column is less than the related intersection, the triple operation can be used. The direction refers to it as a portion of the pivot distance while determining the quickest route from node I to node j using the matrixD_n&S_n

EXAMPLE

Find shortest distance for all-pairs of vertices

-2



Make a matrix of size 5*5, with 5 representing the number of vertices. The numbers I and j are listed for the row and column, respectively. The graph's vertices are I and j.
There is a distance from vertex I to vertex j in each cell D[i][j]. The cell is left as infinity if there is no path connecting the ith and jth vertex.

$D_0 =$

	1	2	3	4	5
1	-	6	∞	7	∞
2	∞	-	5	8	-4
3	∞	-2	-	∞	∞
4	∞	∞	-3	-	9
5	2	∞	7	∞	-

Now select first row and first column

$D_0 =$

	1	2	3	4	5
1	-	6	∞	7	∞
2	∞	-	5	8	-4
3	∞	-2	-	∞	∞
4	∞	∞	-3	-	9
5	2	∞	7	∞	-

Create matrix D 1 now by utilizing matrix D 0. The first row and column's component are left in their current state.Then finding the remaining value by using formula $d_{ij} = \min(d_{ij},\ d_{ik} + d_{kj})$
Where d_{ik}&d_{kj} are corresponding elements of selected row and column.

$d_{23} = \min(5, \infty + \infty) = 5,$

$d_{24} = \min(8\ , 7 + \infty) = 8$

$d_{25} = \min(-4, \infty + \infty) = -4$

$d_{32} = \min(-2, 6 + \infty) = -2$

$d_{34} = \min(\infty, 7 + \infty) = \infty$

$d_{35} = \min(\infty, \infty + \infty) = \infty$

$d_{42} = \min(\infty, 6 + \infty) = \infty,$

$d_{43} = \min(-3\ , \infty + \infty) = -3$

$d_{45} = \min(9, \infty + \infty) = 9$

$d_{52} = \min(\infty, 6 + 2) = 8$

$d_{53} = \min(7, \infty + 2) = 7$

$d_{54} = \min(\infty, 7 + 2) = 9$

column

	1	2	3	4	5
1	-	6	∞	7	∞
2	∞	-	5	8	-4
3	∞	-2	-	∞	∞
4	∞	∞	-3	-	9
5	2	8	7	9	-

5 2 8 7 9 -

=

Now, create a matrix D_2 using matrix D_1. The element in first row and first column are left as it is. Then finding the remaining value by using formula $d_{ij} = \min(d_{ij},\ d_{ik} + d_{kj})$
Where d_{ik}&d_{kj} are corresponding elements of selected row and column.

$d_{13} = \min(\infty, 6+5) = 11,$

$d_{14} = \min(7\ , 6+8) = 7$

$d_{15} = \min(\infty, 6-4) = 2$

$d_{31} = \min(\infty, \infty - 2) = \infty$

$d_{34} = \min(\infty, 8-2) = 6$

$d_{35} = \min(\infty, -4-2) = -6$

$d_{41} = \min(\infty, \infty + \infty) = \infty,$

$d_{43} = \min(-3\ , \infty + 5) = -3$

$d_{45} = \min(9, -4+\infty) = 9$

$d_{51} = \min(2, 8+\infty) = 2$

$d_{53} = \min(7{,}8+5) = 7$

$d_{54} = \min(9{,}8+8) = 9$

=

1	0	-	5	8	-4
2	∞	-2	-	6	-6
3	∞	∞	-3	-	9
4	2	8	7	9	-
5					

Now, create a matrix D_3 using matrix D_2. The element in first row and first column are left as it is. Then finding the remaining value by using formula $d_{ij} = \min(d_{ij},\ d_{ik} + d_{kj})$
Where d_{ik}&d_{kj} are corresponding elements of selected row and column.

$d_{12} = \min(6, 11-2) = 6,$

$d_{14} = \min(7\ , 11+6) = 7$

$d_{15} = \min(2, 11-6) = 2$

$d_{21} = \min(\infty, 11+\infty) = \infty$

$d_{24} = \min(8, 11+6) = 8$

$d_{25} = \min(-4, 11-2) = -4$

$d_{41} = \min(\infty, \infty - 3) = \infty,$

$d_{42} = \min(\infty\ , -2-3) = -5$

$d_{45} = \min(9, -3-6) = -9$

$d_{51} = \min(2, \infty + 7) = 2$

$d_{52} = \min(8, -2+7) = 5$

$d_{54} = \min(9{,}7+6) = 9$

=

1	0	6	5	7	2
2	∞	-	5	8	-4
3	∞	-2	-	6	−6
4	∞	−5	-3	-	-9
5	2	5	7	9	-

Now, create a matrix D_4 using matrix D_3. The element in first row and first column are left as it is. Then finding the remaining value by using formula $d_{ij} = \min(d_{ij},\ d_{ik} + d_{kj})$
Where d_{ik}&d_{kj} are corresponding elements of selected row and column.

$d_{12} = \min(6, 7-5) = 2$

$d_{13} = \min(11\ ,7-3) = 4$

$d_{15} = \min(2,7-9) = -2$

$d_{21} = \min(\infty, 8+\infty) = \infty$

$d_{23} = \min(5,8-3) = 5$

$d_{25} = \min(-4,8-9) = -4$

$d_{31} = \min(\infty, 6+\infty) = \infty$

$d_{32} = \min(-2\ ,6-5) = -2$

$d_{35} = \min(-6,6-9) = -6$

$d_{51} = \min(2, \infty+9) = 2$

$d_{52} = \min(5, -5+9) = 4$

$d_{53} = \min(7,9-3) = 6$

=

	1	2	3	4	5
1	-	∞	∞	∞	2
2	∞	-	5	8	-4
3	∞	-2	-	6	−6
4	∞	−5	-3	-	-9
5	2	4	6	9	-

Now, create a matrix D_5 using matrix D_4. The element in first row and first column are left as it is. Then finding the remaining value by using formula $d_{ij} = \min(d_{ij},\ d_{ik} + d_{kj})$Where d_{ik}&d_{kj} are corresponding elements of selected row and column.

$d_{12} = \min(2, -2+4) = 2$

$d_{13} = \min(4\ ,6-2) = 4$

N $d_{14} = \min(7,9-2) = 7$

$d_{21} = \min(\infty, -4+2) = -2$

$d_{23} = \min(5, -4+6) = 2$

A $d_{24} = \min(8, -4+9) = 5$

$d_{31} = \min(\infty, -6+2) = -4$

$d_{32} = \min(1\ , -6+4) = -2$

$d_{34} = \min(6, -6+9) = 3$

$d_{41} = \min(\infty, -9+2) = -7$

$d_{42} = \min(-5, -9+4) = -5$

$d_{43} = \min(-3, -9+6) = -3$

	1	2	3	4	5
1	-	−2	−4	−7	2
2	−2	-	−4	−7	0
3	−4	−6	-	−9	−2
4	−7	−9	−6	-	−5
5	2	4	6	9	-

1. The Floyd-Warshall algorithm facilitates the inversion of real matrices.
2. It facilitates the analysis of the bipartiteness of the undirected graph.
3. It aids in directed graph shortest path calculation.
4. Various iterations of the Floyd-Warshall algorithm assist in locating a directed graph's transitive closure.
5. This approach aids in identifying the regular expressions that finite automata accept.
6. It assists in identifying graph similarity.
7. The Floyd- Warshall method aids in determining the best route, or the amount of flow between two vertices.

Comparison of three algorithms:

There are several parallels between the Dijkstra algorithm and the Bellman-Ford algorithm. Both of them use the relaxation calculation method, which finds the shortest path by varying the values of D[i] as it travels along the edges and vertices of the graph. In essence, the bellman-ford algorithm can handle the graph with non-negative weight nodes but the Dijkstra algorithm cannot. The Bellman-Ford method, however, is a lot more redundant and inefficient. Any problem involving two points can be resolved using the Floyd-Warshall algorithm.It is more appropriate for use in tiny data sets or when figuring out the shortest route between all vertices. The algorithm is easy to use even though it is useless. The Floyd algorithm wastes redundancy and is the slowest when there are many points and edges.

Conclusion:

The shortest path is the most intriguing research topic. Here, we discuss shortest-path issues. It summarizes how various algorithms have been applied. Bellman Ford in addition to the single-source Dijkstra algorithm is used when there is a negative loop. A dynamic programming algorithm is the Floyd-Warshall algorithm. It locates the two vertices' shortest path. After examining every method, we have come to the conclusion that the Floyd-Warshall algorithm is the most accurate for routing. Floyd-Warshall algorithm gives appropriate time and space for the aforementioned problem. However, it is the most time-consuming and slow method.

REFERENCE:

1. Brendan.H.a(2005). Generalizing Dijkstra's algorithm for solving MDPS. International conference on automated planed and scheduling.
2. Lieberman,F.S.(2001). Introduction to operations research seventh edition.
3. Goldberg ,A.v. (n.d). point to point shortest path algorithms with pre-processing.silicon valley – Microsoft research.
4. Ayunita,Pramana,B, &tajidun, L.(2017). AplikasipencarianRuteterpendekapotek Di kota Kendari MenggunakanAlogorithm Floyd-warshall.
5. Katalog Bps. (2016). Jakarta Dalam Angka 2016. Jakatra Bps provinsi DKI Jakarta.
6. Yusaputra, R.(2013). Aplikasi Mobile pencarianRuteTerpendeklokasifasilitasUmumBerbasis Android menggunakan
7. Pressman, R.S.(2015). Software engineering . A practitioner's Approach, Eighth edition. United states of America : Mcgraw-Hill education.
8. Marseguera.M, and Zio.E. 2000. Genetic algorithms: Theory applications in the safety Domain. Department of Nuclear engineering
9. Pearsons J (1998) Heuristic search in Route finding master's thesis university.
10. Sedgewik R and uitter J S (1986) shortest path in Euclidean Graphs Algorithm.