International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at : www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET2411261

Implementation of Shortest Path Algorithms

Ms. Amruta Navale*, Mrs. Sneha Pawar, Mrs. Bharati Bhamare

Department of Computer Science, Dr. D. Y. Patil ACS College, Pimpri, Pune, Maharashtra, India

ARTICLEINFO

Article History:

Accepted: 15 April 2024
Published: 22 April 2024

Publication Issue :
Volume 11, Issue 2
March-April-2024

Page Number :
392-397

ABSTRACT

An important area of study that models the relationships between items is called graph theory. The shortest path between two objects is one of the most important concepts in graph theory. Many algorithms, such as Dijkstra's Algorithm, Prim's Algorithm, Floyd Warshall Algorithm have been created for this purpose.

Keywords : Dijkstra's Algorithm, Prim's algorithm, Floyd-Warshall Algorithm, Kruskal's Algorithm,

I. INTRODUCTION

Graphs are discrete structures composed of vertices and the nodes that link them. It is attached how networks can be inscribed. Graphs have other properties. In graph theory, we investigated graphs. A graph is composed of vertices and edges. In discrete mathematics, graphs are the focus of the research. There are two sorts of graphs: directed and undirected. GA graph theory is incomplete without the Shortest Path Algorithm, which has several applications in diverse domains. Many mathematicians throughout the world have been working on graph theory. Graph theory and its applications are studied in many domains within mathematics. Graph theory has various real-world applications, including epidemics and pandemics, different types of viruses using bipartite graphs, shortest path, shortest way on a road or network, and many more.

The shortest path on a road map is the challenge of determining which vertices should be represented as uncommon occurrences, with each loaded by the area's distance. There are three types of shortest path or route problems: 1) Single-source shortest path or route; 2) All-pair shortest path or route; and 3) Single-destination shortest path or route problem. To find the shortest path or route from an originating vertex to all remaining vertices, use the single-source shortest path or route. Find the shortest paths between each pair of vertices, as in the all-pair shortest path or route issue. Find the shortest paths from total vertices to a single include the entire destination vertex within a single destination vertex. The goal of these algorithms is to find paths into actual locations on network mapping. There are other factors and variations in algorithms. Cherkassky, Goldberg, and Radzik (1996) could be the ones to start it. Typically, the biggest path—known as the min-delay path—is connected to the shortest path. Additional uses of shortest path issues are being

investigated in the fields of operations research, robotics, transportation, and VLSI design. An easy-to-understand approach for upgrading difficulties is called a greedy algorithm.

What Makes a Graph

Vertices: The basic building blocks of a graph are called vertices. Vertices are sometimes referred to as nodes or vertices. It is possible to label or not label each node or vertex.

Edges: In a graph, edges are used to join two nodes together. In a directed graph, it can be an ordered pair of nodes. Any two nodes can be connected by edges in any way imaginable. There are not any guidelines. Occasionally, arcs are used to refer to edges. It is possible to label or unlabeled any edge.

Sparse graph: A sparse graph is one that has many fewer edges than the expected number of edges.

Dense graph: A dense graph has an edge count that approaches its edge maximum.

Time complexity: The total amount of computer time required to perform an algorithm is referred to as time complexity.

Space complexity: The number of memory regions required to solve a problem is known as space complexity.

Negative weight on a graph: When the total weight of an edge is negative, it is referred to as negative weight.

Negative weight cycle: A negative weight cycle is a rotation with weights that have a non-positive total.

Kruskal's Minimum Spanning Tree (MST)

In Kruskal's algorithm, sort all edges of the provided graph in ascending order. The algorithm then continues to add additional edges and nodes to the MST if the newly added edge does not establish a cycle. It selects the lowest weighted edge first and the highest weighted edge last. As a result, it makes a locally optimum option in each phase to discover the best solution. Hence, this is a greedy algorithm.

Kruskal's procedure to calculate MST

The following are the stages for determining MST using Kruskal's algorithm:

1.Sort the edges in non-decreasing order of weight.
2. Choose the smallest edge. Check to see if it creates a cycle with the spanning tree you've already created. If no cycle is established, include this edge. Otherwise, trash it.
3. Repeat step 2 until there are (V-1) edges in the spanning tree.

Time Complexity: O(E * logE) or O(E * logV)

Applications where Kruskal's method is commonly used:

1. Landing Cables
2. Television Network.
3. Tour Operations.
4. LAN Networks.
5. A network of pipes carrying drinking water or natural gas.
6. An electrical grid.
7. Single-Link Cluster

Using Kruskal's algorithm:

Cable Route	Cost (INR)
A to B	₹2000
A to C	₹1500
A to D	₹1800
B to C	₹2200
B to D	₹2500
C to D	₹2000

1. We start with all cities disconnected.
2. We sort the cable routes based on their costs:

- A-C (₹1500), A-D (₹1800), A-B (₹2000), C-D (₹2000), B-C (₹2200), B-D (₹2500)

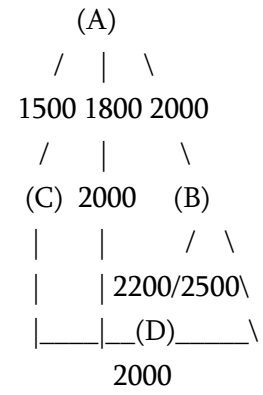
1. We start adding cable routes in ascending order of cost:

- A-C (₹1500) - Added

- A-D (₹1800) - Added
- A-B (₹2000) - Added
- C-D (₹2000) - Creates redundancy with A-C, so skipped
- B-C (₹2200) - Added
- B-D (₹2500) - Added

After this, all cities are connected with minimum total cost

₹1500 + ₹1800 + ₹2000 + ₹2200 + ₹2500 = ₹10000 and no redundant routes are created. This represents the minimum-cost television network.



Prim's Algorithm

The algorithm begins with an empty spanning tree. The goal is to have two sets of vertices. The first set comprises vertices that have already been included in the MST, whereas the second set contains vertices that have yet to be added. At each step, it evaluates all of the edges that link the two sets and selects the lowest weight edge among them. After selecting the edge, it switches its other endpoint to the set containing MST. This algorithm was formed in 1930 by Czech mathematician Vojtech Jarnik.Then; it is retrieved in 1959 by Edsger W. Dijkstra. Also it is known as Jarnik's algorithm.

Time Complexity: O(V2), If the input graph is represented using an adjacency list, then the time complexity of Prim's algorithm can be reduced to O(E * logV) with the help of a binary heap. In this implementation, we are always considering the spanning tree to start from the root of the graph

Applications in which Prim's algorithm is commonly used:

1. Prim's method can solve all of the applications listed in Kruskal's algorithm (in the case of a dense network).
2. A network of roads and rail tracks linking all cities.
3. Creating irrigation canals and installing microwave towers
4. Designing a fiber-optic grid or integrated circuits.
5. The Travelling Salesman Problem.
6. Clustering analysis.
7. Path finding techniques used in artificial intelligence (AI).
8. Game Development.
9. Cognitive science

Let's use the same example with the following costs in Indian Rupees (INR):

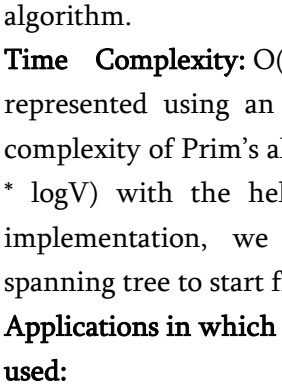
- A to B: ₹2000
- A to C: ₹1500
- A to D: ₹1800
- B to C: ₹2200
- B to D: ₹2500
- C to D: ₹2000

Using Prim's algorithm:

1. **Start with City A**: We start with City A as our starting point.
2. **Grow the Tree**:

- Add the cable route A-C (₹1500) to connect A to C.
- Add the cable route A-D (₹1800) to connect A to D.
- Add the cable route C-D (₹2000) to connect C to D.
- Add the cable route A-B (₹2000) to connect A to B.

Now, all cities are connected, and we have the minimum-cost television network.



Dijkstra's algorithm

The goal is to find the shortest path from the starting point (depot) to each delivery destination, considering the distances between locations and the road network's layout. You can continue updating this table as Dijkstra's algorithm iterates through the nodes, discovering shorter paths and updating distances until all nodes have been visited.

Let's create a sample table to represent the distances from the source node (depot) using Dijkstra's algorithm in the context of route planning for a delivery truck:

Node (Intersection)	Distance from depot (km)	Visited
A	0	Yes
B	∞	No
C	∞	No
D	∞	No

Each row represents an intersection (node) in the road network.

1.The "Node"column lists all the intersections(A, B, C, D).

2.The "Distance from Depot" column represents the distance from the depot (source node) to each intersection. Initially, all distances are set to infinity (∞), except for the depot itself, which has a distance of 0.

3.The "Visited" column indicates whether each intersection has been visited by the algorithm yet. Initially, only the depot is marked as visited (Yes).

As D dijstra's algorithm progresses, the distances from the depot to other intersections will be updated, and the "Visited" column will be updated accordingly. The algorithm will continue to iterate until all intersections have been visited and their shortest paths from the depot have been determined.

Floyd Warshall algorithm

The Floyd Warshall method is an all-pair shortest path method, whereas Dijkstra and Bellman Ford are single-source shortest path algorithms. This approach works with both directed and undirected weighted graphs. However, it does not operate on graphs with negative cycles. It uses Dynamic Programming to check every potential path via every conceivable node in order to find the shortest distance between each pair of nodes.

Steps:

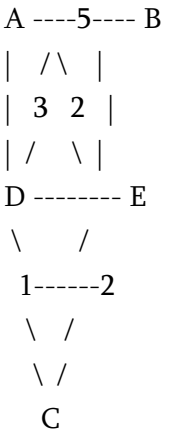
1. First, initialize the solution matrix with the same values as the input graph matrix.
2. Then, update the solution matrix by treating all vertices as intermediates.
3. The goal is to pick all vertices one at a time and update all shortest routes that involve the selected vertex as an intermediate vertex.
4. When we choose vertex number k as an intermediate vertex, we have previously examined vertices {0, 1, 2,.. k-1} as intermediates.
5. For any pair (i, j) of source and destination vertices, there are two possibilities.
6. K is not an intermediate vertex along the shortest path from i to j. We retain dist[i][j] at its current value.
7. K is an intermediate vertex.

Time Complexity: O(V3), where V is the number of vertices in the graph and we run three nested loops each of size V

Auxiliary Space: O(V2), to create a 2-D matrix in order to store the shortest distance for each pair of nodes.

let's take a practical example of network routing using the Floyd-Warshall algorithm.

Imagine you have a network of routers connected by links, and you want to find the shortest path between any two routers in the network. Each link has a certain cost associated with it, representing factors like distance, latency, or congestion.consider a small network with five routers (A, B, C, D, and E) connected like this:



Here, the numbers on the links represent the costs associated with each link.

We can represent this network as an adjacency matrix:

| A | B | C | D | E

A | 0 | 5 | 3 | 1 | ∞

B | 5 | 0 | ∞ | ∞ | 2

C | 3 | ∞ | 0 | ∞ | ∞

D | 1 | ∞ | ∞ | 0 | 2

E | ∞ | 2 | ∞ | 2 | 0

Iteration 1:

Adding Router 1 as intermediate node:

| A | B | C | D | E

A | 0 | 5 | 3 | 1 | 4

B | 5 | 0 | 8 | 6 | 2

C | 3 | 8 | 0 | 4 | 6

D | 1 | 6 | 4 | 0 | 2

E | 4 | 2 | 6 | 2 | 0

Iteration 2:

Adding Router 2 as intermediate node:

| A | B | C | D | E

A | 0 | 5 | 3 | 1 | 4

B | 5 | 0 | 6 | 6 | 2

C | 3 | 6 | 0 | 4 | 6

D | 1 | 6 | 4 | 0 | 2

E | 4 | 2 | 6 | 2 | 0

Iteration 3:

Adding Router 3 as intermediate node:

| A | B | C | D | E

A | 0 | 5 | 3 | 1 | 4

B | 5 | 0 | 6 | 6 | 2

C | 3 | 6 | 0 | 4 | 6

D | 1 | 6 | 4 | 0 | 2

E | 4 | 2 | 6 | 2 | 0

Iteration 4:

Adding Router 4 as intermediate node:

| A | B | C | D | E

A | 0 | 5 | 3 | 1 | 4

B | 5 | 0 | 6 | 6 | 2

C | 3 | 6 | 0 | 4 | 6

D | 1 | 6 | 4 | 0 | 2

E | 4 | 2 | 6 | 2 | 0

Iteration 5:

Adding Router 5 as intermediate node:

| A | B | C | D | E

A | 0 | 5 | 3 | 1 | 4

B | 5 | 0 | 6 | 6 | 2

C | 3 | 6 | 0 | 4 | 6

D | 1 | 6 | 4 | 0 | 2

E | 4 | 2 | 6 | 2 | 0

So, the shortest paths between each pair of routers are:

Shortest path from A to B: 5
Shortest path from A to C: 3
Shortest path from A to D: 1
Shortest path from A to E: 4
Shortest path from B to C: 6
Shortest path from B to D: 6
Shortest path from B to E: 2
Shortest path from C to D: 4
Shortest path from C to E: 6
Shortest path from D to E: 2

II. DISCUSSION

Dijkstra and Floyd Warshall solved every pair's shortest route issue. Prim's , Kruskal's algorithm, dijkstra's provides a robust and efficient solution for the construction of television networks and other network connectivity problems. Its simplicity, scalability, and ability to find the minimum-cost spanning tree make it a valuable tool in infrastructure planning and optimization.

III.CONCLUSION

This paper presents an overview of significant implementation of Shortest Path Algorithms. We compared all shortest route methods to determine the distance between them. We analyzed various research articles on shortest route algorithms and discussed a few key ones in our work. This research detected and comprehended the gap between all of these methods.

IV. REFERENCES

[1]. Prim, R. C. (November 1957), "Shortest connection networks And some generalizations", Bell System Technical Journal, 36 (6): 1389–1401, Bibcode:1957BSTJ...36.1389P, doi:10.1002/j.1538-7305.1957.tb01515.x. 11.

[2]. https://www.geeksforgeeks.org/

[3]. Survey of Shortest Path AlgorithmsSSN: 2321-1067, https://doi.org/10.58443/IJREX.10.11.2022.46-5 Sadhana H.Barkund

[4]. Dijkstra's algorithm". Introduction to Algorithms (Second Ed.). MIT Press and McGraw–Hill. pp. 595–601. ISBN 0-262-03293-7.

[5]. Floyd, Robert W. (June 1962). "Algorithm 97: Shortest Path". Communications of the ACM. 5 (6): 345. Doi:10.1145/367766.368168.

[6]. Kruskal, J. B. (1956). "On the shortest spanning subtree of a graph and the traveling salesman problem ."Proceedings of the American Mathematical Society. 7 (1): 48–50. Doi: 10.1090/S0002-9939-1956-0078686-7.