International Journal for Modern Trends in Science and Technology, 6(12): 479-483, 2020

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST061293
Available online at: http://www.ijmtst.com/vol6issue12.html

Pathfinder Visualizer of Shortest Paths Algorithms

Nishant Kumar[1] | Nidhi Sengar[2]

[1]B-tech. I.T., Maharaja Agrasen Institute of Technology, GGSIPU, New Delhi, India
[2]2Assistant Professor I.T., Maharaja Agrasen Institute of Technology, GGSIPU, New Delhi, India

To Cite this Article
Nishant Kumar and Nidhi Sengar, "Pathfinder Visualizer of Shortest Paths Algorithms", *International Journal for Modern Trends in Science and Technology*, 6(12): 479-483, 2020.

Article Info
Received on 16-November-2020, Revised on 09-December-2020, Accepted on 12-December-2020, Published on 21-December-2020.

ABSTRACT

Visualizations of algorithms contribute to improving computer science education. The process of teaching and learning of algorithms is often complex and hard to understand problem. Visualization is a useful technique for learning in any computer science course. In this paper an e-learning tool for shortest paths algorithms visualization is described. The developed e-learning tool allows creating, editing and saving graph structure and visualizes the algorithm steps execution. It is intended to be used as a supplement to face-to-face instruction or as a stand-alone application. The conceptual applicability of the described e-learning tool is illustrated by implementation of Dijkstra algorithm. The preliminary test results provide evidence of the usability of the e-learning tool and its potential to support students' development of efficient mental models regarding shortest paths algorithms. This e- learning tool is intended to integrate different algorithms for shortest path determination.

KEYWORDS: *Algorithm visualization, Dijkstra algorithm, e-learning tool, shortest path.*

I. INTRODUCTION

Pathfinding or pathing is the plotting, by a computer application, of the shortestroute between two points. It is a more practical variant on solving mazes.This field of research is based heavily on Dijkstra's algorithm for finding theshortest path on a weighted graph.

Pathfinding is closely related to the shortest path problem, within graph theory,which examines how to identify the path that best meets some criteria (shortest,cheapest, fastest, etc) between two points in a large network.At its core, a pathfinding method searches a graph by starting at one vertex and exploring adjacent nodes until the destination node is reached, generally with theintent of finding the cheapest route. Although graph searching methods such as a breadth-first search would find aroute if given enough time, other methods, which "explore" the graph, would tendto reach the destination sooner.An analogy would be a person walking across a room; rather than examiningevery possible route in advance, the person would generally walk in the directionof the destination and only deviate from the path to avoid an obstruction, andmake deviations as minor as possible.

Algorithm :-

Edsger Dijkstra is Dutch. He is one of the big names in computer science. He is known for his handwriting and quotes such as:

- Simplicity is prerequisite for reliability.
- The question of whether machines can think is about as relevant as the question of whether submarines can swim.

Dijkstra's algorithm was created in 1959 by Dutch Computer Scientist Edsger Dijkstra.While employed at the Mathematical Center in Amsterdam, Dijkstra was asked todemonstrate the powers of ARMAC, a sophisticated computer system developed by themathematical Center. Part of his presentation involved illustrating the best way to travelbetween two points and in doing so, the shortest path algorithm was created. It was lateron renamed as Dijkstra's Algorithm in recognition of its creator.In particular, we are reminded that this famous algorithm was strongly inspired byBellman's principle of optimality and thatboth conceptually and technically it constitutesa dynamic programming successive approximation procedure par excellence.

A* Algorithm

In 1964 Nils Nilsson invented a heuristic based approach to increase the speed of Dijkstra's algorithm. This algorithm was called A1.

In 1967 Bertram Raphael made dramatic improvements upon this algorithm, but failed to show optimality. He called this algorithm A2.

Then in 1968 Peter E.Hart introduced an argument that proved A2 was optimal whenusing a consistent heuristic with only minor changes. His proof of the algorithm alsoincluded a section that showed that the new A2 algorithm was the best algorithm possible given the conditions.

He thus named the algorithm in kleene star syntax to be the algorithm that starts with A and includes all possible version number or A*

II. LITERATURE REVIEW

Literature Review is required to take the matter into considerations that can't be cleared in the past researches. Many researchers try to interpret various kind of conclusions and to improve those past results literature review is needed. The present literature serves many varied interesting features, which forms the vital background for the study and conducted a consideration.

An important field of mathematical theory is the mathematical study of the structure of abstract relationships between objects by means of graphs (networks). Although investigating of these constructions can be purely theoretical, they can be used to model pair wise relationships in many real world systems. One of most widely using applications is determination of shortest paths in many practical applications as: maps; robot navigation; texture mapping; typesetting in TeX; urban traffic planning; optimal pipelining of VLSI chips; subroutines in advanced algorithms; telemarketer operator scheduling; routing of telecommunications messages; approximating piecewise linear functions; network routing protocols (OSPF, BGP, RIP); exploiting arbitrage opportunities in currency exchange; optimal truck routing through given traffic congestion pattern.

DATA STRUCTURES

In practice, graphs are usually represented by one of two standard data structures: adjacency lists and adjacency matrices. At a high level, both data structures are arrays indexed by vertices; this requires that each vertex has a unique integer identifier between 1 and V. In a formal sense, these integers are the vertices.

ADJACENCY LISTS

By far the most common data structure for storing graphs is the adjacency list. An adjacency list is an array of lists, each containing the neighbors of one of the vertices (or the out-neighbors if the graph is directed). For undirected graphs, each edge uv is stored twice, once in u's neighbor list and once in v's neighbor list; for directed graphs, each edge uv is stored only once, in the neighbor list of the tail u. For both types of graphs, the overall space required for an adjacency list is O(V + E).

There are several dierent ways to represent these neighbor lists, but the standard implementation uses a simple singly-linked list. The resulting data structure allows us to list the (out-)neighbors of a node v in O(1 + deg(v)) time; just scan v's neighbor list. Similarly, we can determine whether uv is an edge in O(1 + deg(u)) time scanning the neighbor list of u. For undirected graphs, we can improve the time to O(1 + min{deg(u), deg(v)}) by simultaneously scanning the neighbor lists of both u and v, stopping either when we locate the edge or when we fall of the end of a list.

ADJACENCY MATRICES

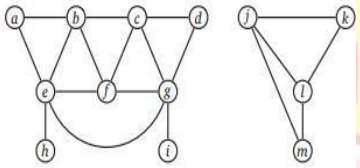
The other standard data structure for graphs is the adjacency matrix, first proposed by Georges Brunel in . The adjacency matrix of a graph G is a V ⇥ V matrix of 0s and 1s, normally represented by a two-dimensional array A[1 .. V, 1 .. V], where each entry indicates whether a particular edge is present in G. Specifically, for all vertices u and v:

if the graph is undirected, then A[u, v] := 1 if and only if uv 2 E, and if the graph is directed, then A[u, v] := 1 if and only if uv 2 E.

For undirected graphs, the adjacency matrix is always symmetric, meaning A[u, v] = A[v, u] for all vertices u and v, because uv and vu are just dierent names for the same edge, and the diagonal entries A[u, u] are all zeros. For directed graphs, the adjacency matrix may or may not be symmetric, and the diagonal entries may or may not be zero.

Given an adjacency matrix, we can decide in ⇥ (1) time whether two vertices are connected by an edge just by looking in the appropriate slot in the matrix. We can also list all the neighbors of a vertex in
⇥ (V) time by scanning the corresponding row (or column). This running time is optimal in the worst case, but even if a vertex has few neighbors, we still have to scan the entire row to find them all.
Similarly, adjacency matrices require ⇥ (V2) space, regardless of how many edges the graph actually has, so they are only space-evident for very dense graphs.

	a	b	c	d	e	f	g	h	i	j	k	l	m
a	0	1	0	0	1	0	0	0	0	0	0	0	0
b	1	0	1	0	1	1	0	0	0	0	0	0	0
c	0	1	0	1	0	1	1	0	0	0	0	0	0
d	0	0	1	0	0	0	1	0	0	0	0	0	0
e	1	1	0	0	0	1	1	1	0	0	0	0	0
f	0	1	1	0	1	0	1	0	0	0	0	0	0
g	0	0	1	1	1	1	0	0	1	0	0	0	0
h	0	0	0	0	1	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	1	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	0	1	1	1
k	0	0	0	0	0	0	0	0	0	1	0	1	0
l	0	0	0	0	0	0	0	0	0	1	1	0	1
m	0	0	0	0	0	0	0	0	0	1	0	1	0



III. METHODOLOGY

In this section, the overall working of the project has been described. How the project started and how the project works and how the various phases of project were carried out and the challenges faced at each level.
What does the project do?
At its core, a pathfinding algorithm seeks to find the shortest path between two points. This project visualizes various pathfinding algorithms in action, and more!
All of the algorithms on this project are adapted for a 2D grid, where 90 degree turns have a "cost" of 1 and movements from a node to another have a "cost" of 1.
Picking an Algorithm
Choose an algorithm from the "Algorithms" drop-down menu. Note that some algorithms are unweighted, while others are weighted. Unweighted algorithms do not take turns or weight nodes into account, whereas weighted ones do. Additionally, not all algorithms guarantee the shortest path.
Meet the algorithms
Dijkstra's Algorithm :The father of pathfinding algorithms; guarantees the shortest path.
Greedy Best-first Search (weighted): A faster, more heuristic-heavy version of A*; does not guarantee the shortest path.
Breath-first Search (unweighted): A great algorithm; guarantees the shortest path.
Depth-first Search (unweighted): A very bad algorithm for pathfinding; does not guarantee the shortest path.
Adding walls
Click on the grid to add a wall.
Walls are impenetrable, meaning that a path cannot cross through them.
Visualizing and more Use the navbar buttons to visualize algorithms and to do other stuff!
You can clear the current path, clear walls and weights, clear the entire board, and adjust the visualization speed, all from the navbar. If you want to access this tutorial again, click on "Pathfinding Visualizer" in the top left corner of your screen.
Objectives of the project:

- It can be used as a E learning tool to understand Algorithms.
- It is used in finding Shortest Path.
- It is used in the telephone network.
- It is used in IP routing to find Open shortest Path First.
- It is used in geographical Maps to find locations of Map which refers to vertices of graph.
- We can make a GPS system which will guide you to the locations.
- Search engine crawlers are used BFS to build index. Starting from source page, it finds all links in it to get new pages.
- In peer-to-peer network like bit-torrent, BFS is used to find all neighbor nodes.
- As users of wireless technology, people demand high data rates beyond GigaBytes per second for Voice, Video and other applications. There are many standards to achieve data rates beyond GB/s. One of the standards is MIMO(Multi input Multi output).MIMO employs K-best Algorithm(which isa Breadth-First Search algorithm) to find the shortest partial euclidian distances.

Phases of the project
The development of the project has been carved out in SixPhases. These phases include all the steps of the project, beginning from data collection and

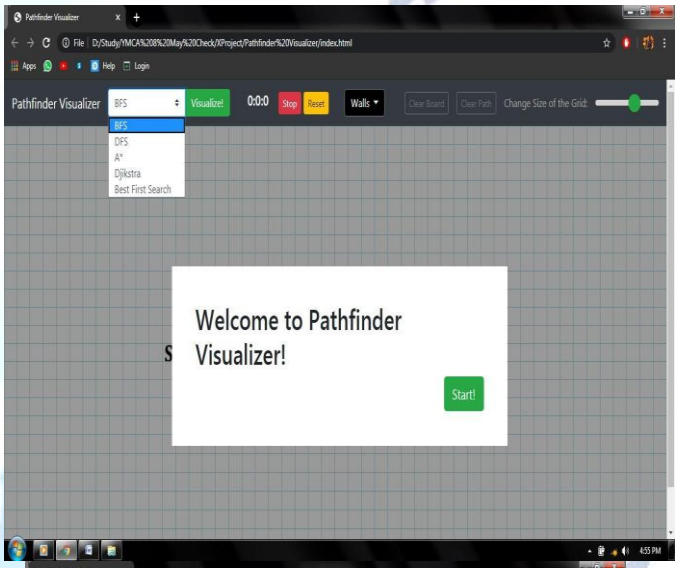
processing to the output for the user. The Six phases are:

1. Building of Graph Matrix.
2. Added Walls and Event listeners.
3. Embed the Graph Algorithms.
4. Integrated the Path finding Functionality.
5. Improved the Design and UI.
6. Added the Timer Functionality

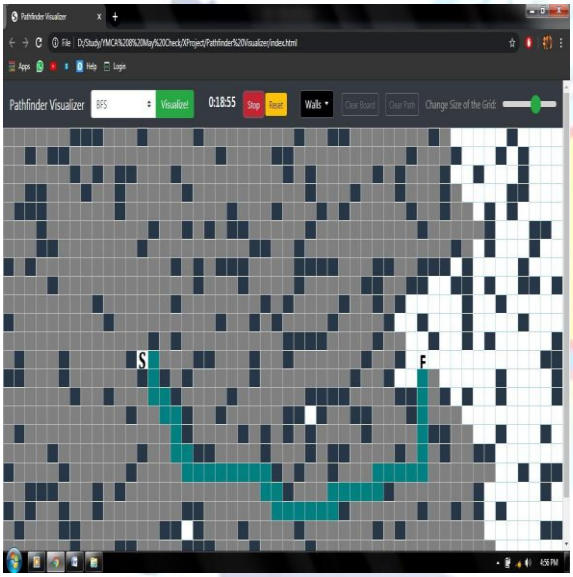
After all these phases the project is completely ready for the user to use. Each Phase has been discussed in detail from here on onwards, letting to a complete understanding of the project

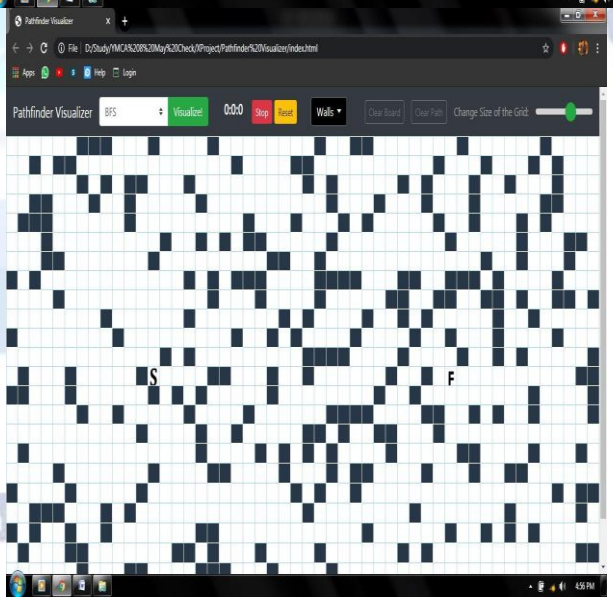
IV. RESULTS

FirstlyClick on Start &Select an Algorithm

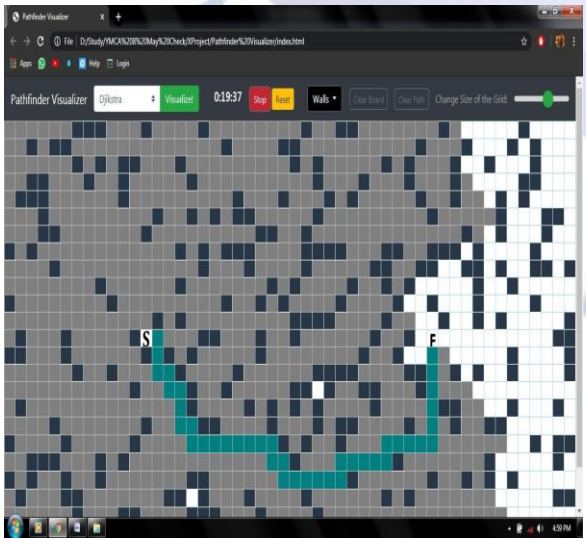


Insert Walls on the Board Grid 1





Visualize Dijkstra's Algorithm



Visualize BFS

Performance Analysis & Comparison

Table

Algorithm	Time(second: millisecond)
BFS	18:55
Dijkstra's Algorithm	19:37

V. CONCLUSION

With the completion of this project, we have successfully achieved our objective of our project is to embed Graph Path Finding with Visualization and Comparing their performance.

As is the case with most other teaching areas, there has been a significant gap between the theory and practical understanding of algorithms realization. This is true also for shortest paths algorithms and in particular for Dijkstra algorithm.

The main goal of the project is to use it from operations research educators and students for teaching and studying the existing known combinatorial graph algorithms.

The main idea of the system is to provide an integrated educational environment for both instructors and students to facilitate the learning process in efficient way.

To conclude, we have learnt a lot of things working under this project. We are also thankful to our mentor and supervisor for their efforts in the learning process.

VI. REFERENCES

1.https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
2.https://medium.com/swlh/pathfinding-algorithms-6c0d4febe8fd#:~:text=What%20are%20Pathfinding%20Algorithms%3F,based%20on%20some%20predefined%20criteria.
3.https://en.wikipedia.org/wiki/Pathfinding
4.https://www.w3schools.com/js/
5.https://www.meta-chart.com/histogram
6.https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
7.https://www.geeksforgeeks.org/a-search-algorithm/
8.https://www.researchgate.net/publication/282488307_Pathfinding_Algorithm_Efficiency_Analysis_in_2D_Grid
9.http://tesi.fabio.web.cs.unibo.it/twiki/pub/Tesi/DocumentiRitenuti Utili/p80-ardito.pdf
10.Ahuja R. K., Magnanti, T. L. & Orlin, J. B. (1993). Network Flows: Theory, Algorithms and Applications. Englewood Cliffs, NJ: Prentice Hall.
11.Dijkstra, E. W. (1959). A Note on Two Problems in Connection with Graphs. Numeriche Mathematik, 1, 269-271.
12.Fouh E., Akbar M. & Shaffer C. A. (2012). The Role of Visualization in Computer Science Education. Computers in the Schools, 29(1-2), 95-117.
13. Roles J.A. & ElAarag H. (2013). A Smoothest Path algorithm and its visualization tool. Southeastcon, In Proc. of IEEE, DOI: 10.1109/SECON.2013.6567453.

14. Paramythis A., Loidl S., Mühlbacher J. R., & Sonntag M. (2005). A Framework for Uniformly Visualizing and Interacting with Algorithms. In Montgomerie, T.C., & Parker E-Learning, J.R. (Eds.), In Proc. IASTED Conf. on Education and Technology (ICET 2005), Calgary, Alberta, Canada, 2-6 July 2005, pp. 28- 33.