IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

Path Finding Visualisation in Mazes Using Various Algorithms

Rohit Kumar[1], K. Sharath[2]

[1]Student, Dept. of MCA, Bangalore Institute of Technology, Bengaluru, India.

[2]Asst. Prof., Dept. of MCA, Bangalore Institute of Technology, Bengaluru, India.

Abstract: A maze always has a way out but finding its way out or finding the destination in the maze is always a challenging task. There are many algorithms that can be used to search the destination node in the maze but which algorithm is going to find the destination node faster. This is what we are going to survey in this paper using some visualization and comparing the algorithms one by one. Some algorithms are fast but cannot provide the shortest route and some provide the shortest route but are really slow. The maze algorithm is first processed over the grid. A specific path finding algorithm is later used to get the shortest path from source node to destination node. The "10X10" grid is used here to learn the functionality of all path finding algorithms.

Keywords: Path Finding Algorithms, uninformed, BFS, DFS, A algorithm.

I. INTRODUCTION

This project is made to help students to understand working of algorithms by the means of visualisation. Path finder visualizer demonstrates algorithms like Dijkstra's, A search, Breath-first search, Depth-first Search, Swarm Algorithm, Greedy Best-first search, convergent swarm algorithm, and Bidirectional swarm algorithm. One source node and one destination node will be given and walls and weights can be created by the user accordingly. User can also add checkpoints i.e., like a pre-destination. In the below figure 1, we have a random maze in a 10x10 grid with a source node and a destination node. There are several ways to get to the destination node but we need the shortest one. We applied the Dijkstra's algorithm for this purpose as it guarantees the shortest path. The left image shows the initial state (before algorithm is applied) whereas right image shows final state (after algorithm is applied). The green colour path indicates the shortest path possible. We can also see the change in the colour of grid blocks, those are the visited nodes and some nodes are unreachable and are not visited.

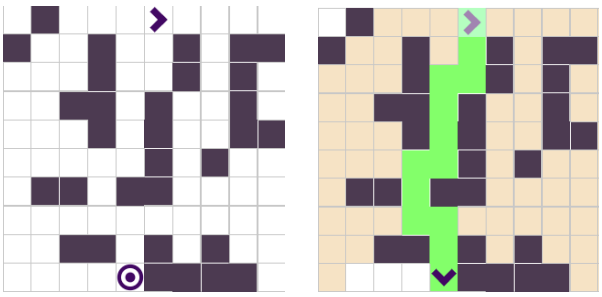

Fig. 1

In this paper, we are mostly focusing on path finding algorithms but we have also applied algorithms to make mazes on the grid. We have built 6 pre-defined maze creation algorithms namely, recursive division, recursive division (vertical skew), recursive division (horizontal skew), basic weight maze, simple stair pattern and basic random maze. Below are the figures showing different mazes.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

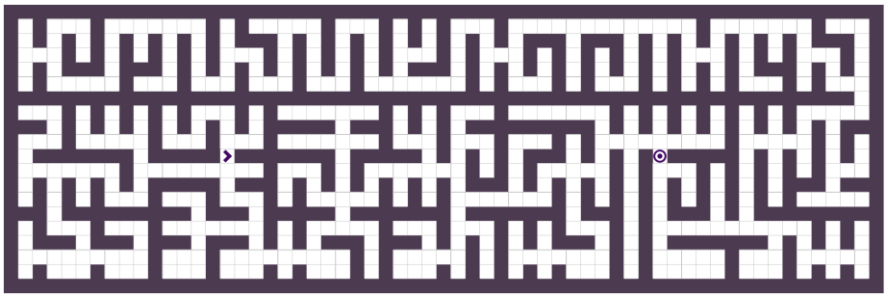

Fig. 2 recursive division

Fig. 3 recursive division (vertical skew)

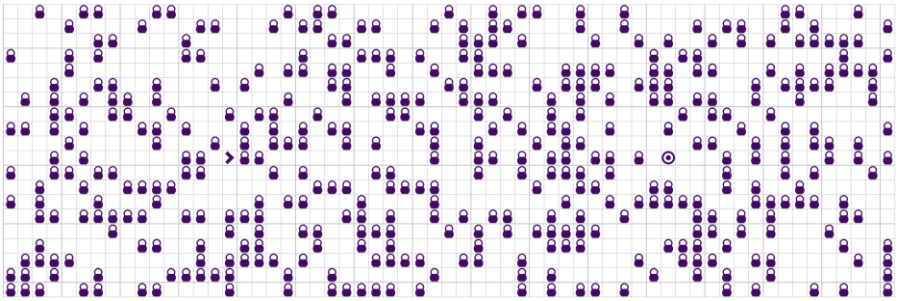

Fig. 4 basic weight maze

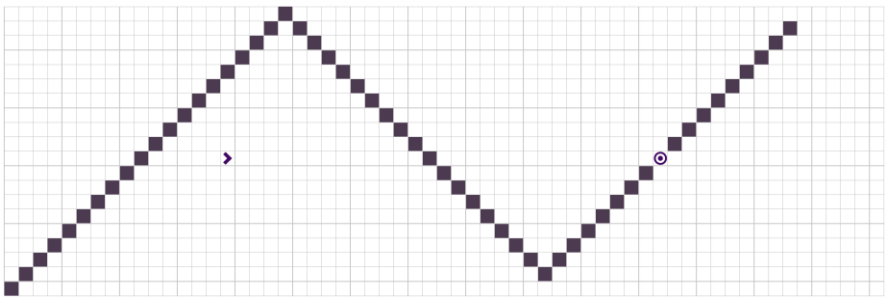

Fig. 5 simple stair pattern

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

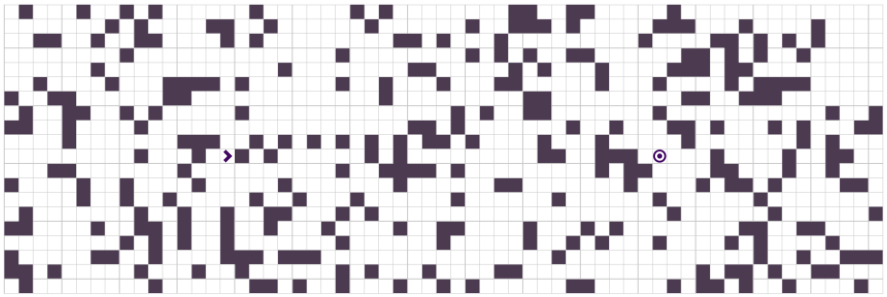

Fig. 6 basic random maze

II. LITERATURE SURVEY

This section includes the abstracts provided by their authors regarding the path finding in maze.

M.O.A. Aqel and his team (2017) [2] used breadth-first search which is one the most Advanced search algorithm to find solutions of mazes. In this they have used a similar kind of pattern in which the blocks (nodes) are distributed in several levels namely, level 0, level 1, level 2, and so on. The level 0 nodes are first queued and visited in the FIFO way. This method works best when maze is of limited size making the search space limited. The time will be proportional to the number of blocks (nodes) on the grid thus increasing the size of the maze is not a great idea.

N.Hazim and his team (2016) [3] used A algorithm in their game strategies to find the shortest path. A search is far better than any other algorithm like BFS or DFS that searches blindly and traverse the nodes until they find the destination node. But A algorithm has its own brain. It covers the distance in a straight line in a heuristic manner to expand nodes. It knows how close the destination node is and keeps searching in that area and adjacent nodes until it finds its way to the destination node. It is better than other blind search algorithms and can be used in large size mazes.

S.J. Russell (2018) [4] discussed diversity of techniques that can be applied to graph data structures to find a shortest path in the maze. There are techniques such a, iterative deepening, depth limited search and other updated and powerful search strategies which can be used in maze problems to find shortest paths from source node to destination node. There are algorithms that can be used over both source node and destination node. Search begins from both nodes and they try to meet a common node which is difficult but it works twice fast and is best for limited grid size. Bi-directional swarm algorithm can be used in this case.

III. WORKING METHODOLOGY

We have created nodes as blocks of the grid. Each block is of 25x25 pixels. Keeping the size of blocks small provides better performance. From the source node the path finding algorithm can go in any direction, it can go up, down, left or right (depending upon the algorithm). Some algorithms like Dijkstra's traverse in every direction equally and keep visiting the nodes until they find the destination node. If the block is white in colour, then it means it is unvisited, if its colour is beige then it means it is visited and if the colour of the block is purple then that block is a wall. Algorithms cannot go through the walls; they have to search a way around them. Below figure 7 visualizes the above details.

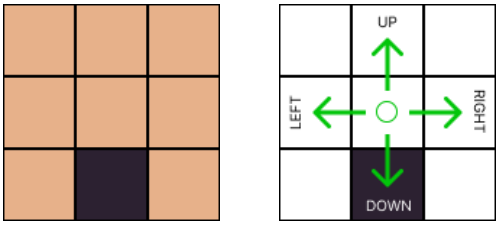

Fig. 7

There are three groups in which every path finding algorithms comes over.

1) If only one path is there

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

If only one way is there to the destination node, then there lies no idea about the shortest path. The algorithm will visit the maze until they find the way to the destination node as all other ways will be closed with wall blocks.

2) No path is there

In this case, the path finding algorithm will check the whole maze and will terminate if it's not able to find any way.

3)Multiple paths are available

In this case, the path finding algorithm will explore the maze as it hits the source node, then it will traverse using the shortest path from all the paths. In Dijkstra's algorithm visualisation takes place in all four directions and it keeps multiplying until it hits the source node. Then from those visited nodes it will find the shortest path to the source as shown in the figure 8.

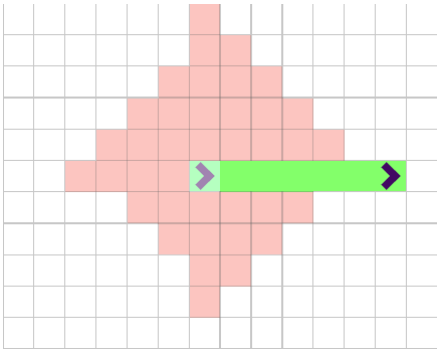

Fig. 8

IV. PATH FINDING ALGORITHMS

This part is compromised with various path finding algorithms that are used for path finding in maze. Path finding algorithms falls in two categories i.e., Informed and Uninformed search strategies [4]. The two sections are:

A. Informed Search Technique

Informed search techniques use more information about specific problems perhaps finding a solution in an effective way. Below is the description of information search algorithm.

1) A Algorithm

The A algorithm sounds new to most of the students but it is one of the best pathfinding algorithm. It uses heuristics to guarantee the shortest path much faster than Dijkstra's Algorithm. It is like it has its own brain. It covers the distance in a straight line in a heuristic manner to expand nodes. It knows how close the destination node is and keeps searching in that area and adjacent nodes until it finds its way to the destination node. It is better than other blind search algorithms and can be used in large size mazes. A algorithm extends the following node (N) for the below function that gives a minimum value.

$$fx(N) = gx(N) + hx(N)$$

Here gx(N) is the weight to traverse from the beginning to the destination node area and hx(N) is a heuristic function which derives the problem, so it will be different for each problem. In the case of the maze, the heuristic used is straight path from the distance of the destination. With more heuristic details, A outclasses all uninformed search algorithms and therefore making it best to solve a maze. Below Figure 9 shows the A algorithm in action.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

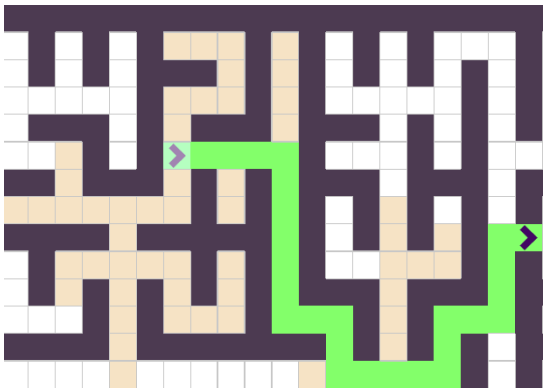

Fig. 9

B. Uninformed Search Strategies

1) Depth-First Search

The Goal of this algorithm is to first traverse deep inside the grid until it reaches the boundary node. The algorithm will work until it finds his destination node or hits the dead-end boundary. In the former case it would be the solution to a problem that may not be the shortest path. The algorithm will use back-tracking and will find another boundary node and repeats the same process until the destination node is found or no remaining node to check. Such the algorithm may seem ready for a maze solution problem but may be some area remain untouched or un-traversed which will weaken the obtained solution path and will no longer be usable. The solution path could be optimal by a close adjustment i.e., keep searching until all nodes are scanned, finally taking a minimum one in termination. The time Complexity of DFS algorithm is O (m + n) where m and n are common variables. The depth-first search algorithm is best if the maze has only one path.

Note: the nodes should be arranged in chronological order. The below figure 5 demonstrates DFS why it explores other nodes and if more than one path is their even then it does not provide a optimal path. Therefore, it is not recommended for pathfinding.

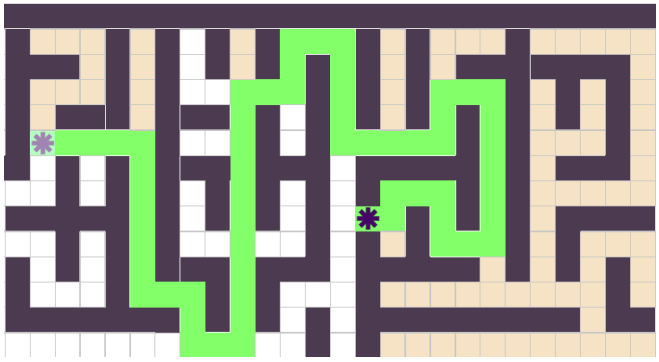

Fig. 10

2) Breadth-First Search

The breadth-first search algorithm searching pattern is same as Dijkstra's but we cannot add weight nodes in this. It is a great algorithm and guarantees the shortest path. The blocks (nodes) are distributed in several levels namely, level 0, level 1, level 2, and so on. The level 0 nodes are first queued and visited in the FIFO way. This method works best when maze is of limited size making the search space limited. The time will be proportional to the number of blocks (nodes) on the grid thus increasing the size of the maze is not a great idea. The breadth-first search algorithm has a marvellous feature that it distributes the nodes and names them by a integer which is the minimum distance from the source node. As you can see in Figure 6.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

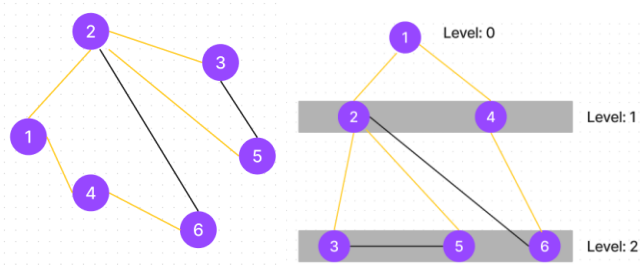

Fig. 11

Above you can see two graphs (Fig. 11) the left side one is not distributed and the right one is distributed level wise. The level wise one is obtained after breadth-first search. The yellow edges are used to traverse from one node to another and the black lines are used to show that those paths are ignored. Levelling is a crucial part in BFS, they are actual length from the starting node. The BFS path finding algorithm performs really well in small maze scenario meaning we need to keep the grid small as much as possible to see the best results. It guarantees the shortest path every time. Below figure 12 shows the same source and destination node as DFS figure but it shows the shortest path.

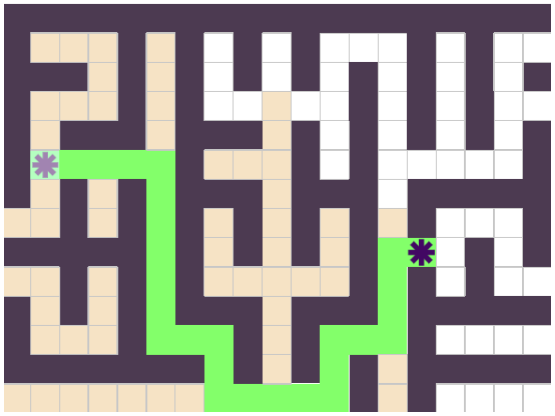

Fig. 12

3) Dijkstra's Algorithm

As we all know Dijkstra's algorithm is the father of pathfinding algorithms and it guarantees the shortest path. It our visualisation we can see it starts finding the destination equally in every direction. It searches for the destination node in every direction. And as it finds the destination it takes the shortest route possible. In the below figure 13 we have used a recursive division maze and used the Dijkstra's algorithm to find the shortest path in our software.

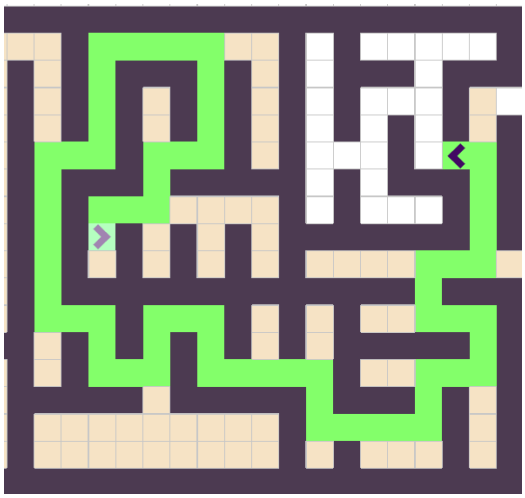

Fig. 13

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96130

V. TABLE OF COMPARISON

Algorithm	Optimal	Performance	Finds source
A Search	Yes	Very High	Yes
Depth –First Search	No	Low	Yes
Breadth-First Search	Yes	High	Yes
Dijkstra's	Yes	Medium	Yes

VI. CONCLUSION

The Path finding algorithms are better and much optimal approach to find destination node in a maze rather than using other algorithms like wall follower. The A algorithm is the best algorithm to find the destination node in a maze. It is most optimal and fast. Its just like it has a brain of its own. Therefore, it is widely used. The DFS took the longest time to find the destination node. It traverses through a large number of nodes before finding the destination node hence, proving to be least optimal. On the other hand, BFS algorithm was quick and optimal. The Dijkstra's algorithm took medium amount of time but provided us the shortest path every time.

At last, I can conclude that this project will help the students to learn about the algorithms and their visualization. By watching how algorithm works one can define the differences and speed index of the algorithm easily.

REFERENCES

[1] JavaScript reference and documentation retrieved from https://devdocs.io/javascript/
[2] A. Issa, M.O.A. Aqel, M. Khdair, M. Abubaker, M. ElHabbash and M. Massoud, "Intelligent Maze Solving Robot Based On Image Processing and Graph Theory," 2017 International Conference on Promising Electronic Technologies (ICPET), page 49-53, October 2017
[3] Nawaf Hazim Barnouti, Sinan Sameer Mahmood Al-Dabbagh and Mustafa Abdul Sahib Naser, "Pathfinding in Strategy Games and Maze Solving Using A Search Algorithm," Journal of Computer and Communications Vol.04 No.11(2016), Article ID:70460,11 pages
[4] P. Norvig and S.J. Russell "Artificial Intelligence: A Modern Approach," 3rd Edition, Pearson India, page 64-108, 2018
[5] N. Garg, "Lecture - 25 Data Structures for Graphs [Video file]", September 24,2008. Retrieved from https://www.youtube.com/watch?v=hk5rQs7TQ7E
[6] M. Chandra Wijaya, S. Tjiharjadi, and E. Setiawan, "Optimization Maze Robot Using A and Flood Fill Algorithm," International Journal of Mechanical Engineering and Robotics Research Vol. 6, No. 5, September 2017
[7] C.E. Leiserson, T.H. Coremen, R.L. Rivest and C. Stein, "Introduction to Algorithms", Third edition, Prentice Hall of India, page 587-748, 2009
[8] Silvester Dian Handy Permana, Ketut Bayu Yogha Bintoro, Budi Arifitama and Ade Syahputra, "Comparative Analysis of Pathfinding Algorithms A , Dijkstra, and BFS on Maze Runner Game" International Journal Of Information System & Technology Vol. 1, No. 2, (2018), pp. 1-8
[9] Xiao Cui and Hao Shi, "A-based Pathfinding in Modern Computer Games" VOL.11 No.1, January 2011
[10] Bi Yu chen, Chaoyang Shi and Shujin Xiang, "Most reliable path-finding algorithm for maximizing on-time arrival probability" Pages 248-264 | Received 01 Jul 2015, accepted 21 Mar 2016, Published online: 13 Apr 2016
[11] Ganesh Khekare, Pushpneel Verma and Seema Raut, "The Optimal Path Finding Algorithm Based on Reinforcement Learning" International Journal of Software Science and Computational Intelligence (IJSSCI) 12(4)
[12] J. Kennedy and R.C. Eberhart (1997), "A discrete binary version of the particle swarm algorithm" IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation.
[13] Baruch Awerbuch, "A new distributed Depth-First-Search algorithm" Volume 20, Issue 3, 8 April 1985, Pages 147-150
[14] V. Nageshwara Rao & Vipin Kumar, "Parallel depth first search. Part I. Implementation" International Journal of Parallel Programming volume 16, pages479–499 (1987)
[15] Dexter C. Kozen, "The Design and Analysis of Algorithms" pp 19–24
[16] Zeev Colin and Shlomi Dolev, "Self-stabilizing depth-first search" Volume 49, Issue 6, 22 March 1994, Pages 297-301