Pranav Balaji
Rishi Dinesh
Sruthi Srinivasan

A Comparative Study of Various Pathfinding Algorithms

A project poster submitted to Dr. PATTABIRAMAN V - School of Computer Science and Engineering in partial fulfilment of the requirements for the course of CSE2003 – DATA STRUCTURES AND ALGORITHMS

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

ABSTARCT

Pathfinding algorithms find their application in several domains such as google maps, satellite navigation systems and outing packets over the internet. Pathfinding algorithms require a search technique to get the fastest route, efficient, and shortest time. This paper presents a comparative analysis of some of the most popular path finding algorithms such as DFS,BFS, A* and Dijkstra. These algorithms were implemented and visualized in python to find the shortest path between the starting and ending point of a maze consisting of obstacles. Two maze generation algorithms- Recursive division and DFS maze were used to generate the maze and these two algorithms have also been highlighted in this paper. This paper describes the working of the path finding algorithms and subsequently attempts to compare the efficiency of the four algorithms based on metrics such as time taken, number of cells considered, cost of path and reachability.

ALGORITHMS

The four algorithms used for the visualization and comparison are

- Depth-First Search
- Breadth-First Search
- Dijkstra's algorithm
- A* Search algorithm

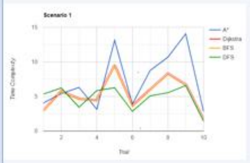
The two automatic maze generation algorithms used in this project are the Recursive Division Algorithm and Depth first Search Maze Algorithm.

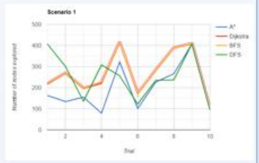
COMPARATIVE ANALYSIS

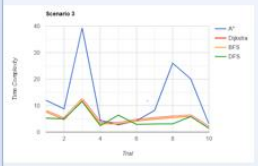
An experiment with a series of trials was conducted to analyze the four different pathfinding algorithms. The experiment was designed in the following manner: Four mazes (or scenarios) were generated, two using the recursive division algorithm and two using the DFS algorithm. For each scenario, a series of 10 trials were conducted to collect data and build observation tables.

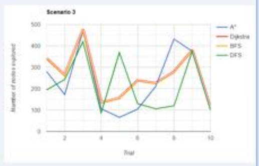
For every trial under one scenario, the parameters varied to get different results were the starting point and the ending point. Four different metrics were used to analyze these algorithms – time complexity (M1), number of nodes considered (M2), final path length (M3) and reachability (M4).

Shown below is a sample of the results of two out of the four scenarios generated, one using Recursive Division (Scenario1) and one using DFS (Scenario3).









INFERENCE

It was inferred that the A* algorithm always guaranteed the shortest path (as indicated by M3). Another observation that was be made was that A* explored minimum number of nodes while finding the shortest path (as indicated by M2). However, it could be seen that the time taken to find the shortest path was high compared to the other three algorithms, especially in mazes generated using DFS (as indicated by M1). This could be attributed to the fact that the speed of execution of A* is heavily dependent on the accuracy and quality of the heuristic function used.

The Dijkstra algorithm, much like the A* algorithm, guaranteed the shortest path in any case (as indicated by M3). However, unlike A*, it explored a lot of nodes before determining the shortest path (as indicated by M2). It had a fairly good time complexity (as indicated by M1) owing to the fact that it is a greedy algorithm.

It could be observed that the results of the Breadth First Search were very close to that of the Dijkstra algorithm. This was a result of the fact that all the nodes in the maze were unweighted. In an unweighted graph, both Dijkstra and BFS work with more or less the same principle. In fact, Breadth-first search can be viewed as a special-case of Dijkstra's algorithm on unweighted graphs, where the priority queue degenerates into a FIFO queue. Dijkstra is more efficient than BFS only in situations with weighted graphs.

The Depth First Search had the least amount of running time out of all the four algorithms. This verified the fact that DFS is more efficient than BFS (in terms of speed and number of nodes explored). It is also known that DFS works better than BFS when the destination is far away from source (which was the case in many of the conducted trials). However, it did not always guarantee the shortest path as indicated by the table shown here.

OBSERVATION SUMMARY

Scenario	A* Algorithm			Dijkstra's Algorithm			Breadth First Search			Depth First Search		
#	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
1	7.22	195	84	5.25	265	84	5.24	265	84	4.86	250	84
2	5.57	144	64	5.35	267	64	5.33	267	64	4.34	220	64
3	12.88	232	95	5.24	256	95	5.23	257	95	4.71	215	108
4	8.36	164	59	4.43	237	59	4.42	238	59	5.60	304	70

CONCLUSION

The results of the comparative analysis showed how the different pathfinding algorithms worked in different maze settings. It was observed that A* always produced the shortest path by exploring a minimum number of nodes, making it the most reliable of all four algorithms. It, however, did not have an optimal running time owing to a poor heuristic function. The Dijkstra algorithm, being the abridged version of A*, displayed a similar performance. Due to the lack of a heuristic function to guide its search, the number of nodes explored were comparatively high. It, however, triumphed where A* failed by maintaining its running time even in DFS generated maze settings.

The striking similarities between the performance results of the Dijkstra algorithm and the Breadth First Search proved the fact they are indeed based on the same concepts when tested in unweighted maze settings. The Depth First Search, despite using an uninformed search strategy, always managed to find a path between the source and destination in very little time. It, however, explored a lot of nodes to achieve this and did not always guarantee the shortest path.