Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

An Investigation into Maze Generation And Pathfinding Algorithms

Author: Rhys Hughes (2116435) Supervisor: Gordon Dickers

School of Applied Computing

Conclusions

Depth-first search was significantly faster than Breadth-first search at solving mazes.

Depth-first search was most efficient on Sidewinder and least efficient on Randomised Kruskals

Breadth-first search was most efficient on Recursive Division and least efficient on Randomised Kruskals.

Introduction

This project attempts to investigate the differences between breadth-first search and depth-first search in the context of solving mazes. To test the two search algorithms, 3 maze generation algorithms were chosen, these being Randomised Kruskals, Recursive Division, and Sidewinder. The search algorithms were timed as they solved different size mazes.

Aims and Objectives

To investigate the time taken for depth-first search and breadth-first search to solve different maze algorithms.

- Research into different search and maze algorithms.
- Investigate how different maze algorithms affect solving times
- Create a program that can generate mazes using Randomised Kruskals, Recursive Division, and Sidewinder
- Create a program that can solve the mazes generated using Depth-first and Breadth-first search algorithms.

Research

Breadth-first search:

Breadth-first search is an uninformed search strategy meaning the only information it has about any given node is whether it is the goal state or not. Theoretically all uninformed search algorithms have the same time complexity of O(V+E) where V is vertices and E is edges. It is an algorithm that first expands the root node and then it's successors after. [1] This means the shallowest node is next to be expanded meaning the queue is FIFO (First in first out). It is guaranteed to find the shortest route to the exit of a maze.

Depth-first search:

Depth-first search is also an uninformed search strategy. It differs by expanding the lowest successor first. [2] This means the queue is LIFO (Last in first out). It is not guaranteed to find the shortest path

Randomised Kruskal:

Based off of Kruskal's minimum spanning tree algorithm [3]. Initially all cells are their own set with all possible edges shuffled. [4] Algorithm goes through shuffled edges connecting the cells together and adds them to the same list.

Recursive Division:

Splits the area into two by adding a walls, leaves one point on the wall blank to allow a passage [4]. Continues until desired resolution

Sidewinder:

Either goes east or north. [4] If it goes east it removes the wall and adds the cells together into a "run". If it goes north, one random cell in the run carves north. Then it resets the "run".

Hypotheses and testing

Primary hypothesis is that breadth-first search is more efficient. The secondary hypotheses are that breadth-first search is more efficient on Recursive division and least efficient on Sidewinder, and depth-first search is more efficient on Recursive Division and least efficient on Randomised Kruskals.

To test thes, each maze algorithm will generate 20 mazes each of 4 different sizes 25x25, 50x50, 75x75, and 100x100 and both search algorithms will be timed in milliseconds to 4 decimal places.

Findings

Depth-first search was significantly more efficient than Breadth-first search on all maze sizes.

25x25: Depth-first search was most efficient on Recursive Division and least efficient on Sidewinder. Breadth-first search was most efficient on Recursive Division and least efficient on Sidewinder.

50x50: Depth-first search was most efficient on Sidewinder and least efficient on Kruskals. Breadth-first search was most efficient on Recursive Division and least efficient on Kruskals.

75x75: Both depth-first and breadth-first were most efficient on Sidewinder and least efficient on Kruskals.

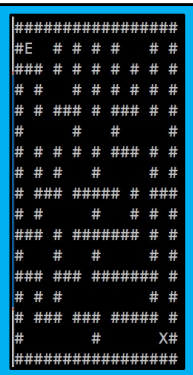
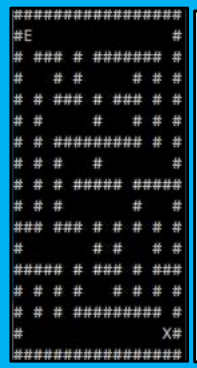
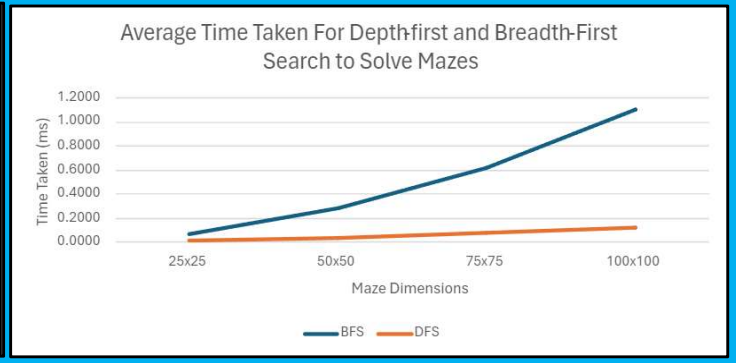
100x100: Depth-first search was most efficient on Sidewinder and least efficient on Kruskals. Breadth-first search was most efficient on Recursive Division, and least efficient on Kruskals

Both the primary and secondary hypotheses were false. On every size maze the quickest breath-first search was slower than the slowest depth-first search result.

Literature cited

[1] S. Russell and P. Norvig, Artificial Intelligence A Modern Approach Third Edition, Upper Saddle River: Pearson, 2016

[2] J. Gross and J. Yellen, Graph Theory and Its Applications Second Edition, Boca Raton: Taylor and Francis Group, 2006, pp. 170-176.

[3] J. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem".

[4] W. Pullen, "Maze Classification," 31 July 2022. [Online]. Available: https://www.astrolog.org/labyrnth/algrithm.htm. [Accessed 15 February 2024]