CS231

11/27/23

Using Priority Queues and Heaps to Solve a Maze (Project 7 Report)

Abstract

In our project, we tried three ways to solve mazes: Depth-First Search (DFS), Breadth-First Search (BFS), and A*. We paid special attention to A* and used a PriorityQueue to find the best paths. We looked at how obstacles in the maze affected reaching the goal and studied the paths and cells explored by each method. The results, shown in tables, tell us how well these methods work for solving mazes.

Exploration

The probability of finding the target given different densities.

Density	Probability of Finding Target (%)
0	100
0.1	100
0.2	80
0.3	30
0.4	10
0.5	0
0.6	0
0.7	0
0.8	0
0.9	0

1.0	0

Lengths of the paths found by different search algorithms.

	BFS	DFS	A*
Length	29	45	29

The average number of cells explored by each search algorithm.

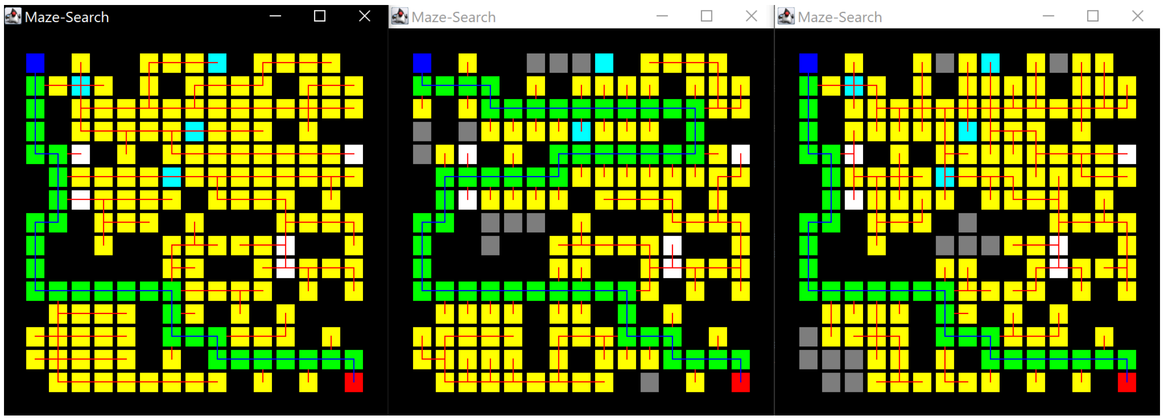
	BFS	DFS	A*
# of Cells	141	93	74

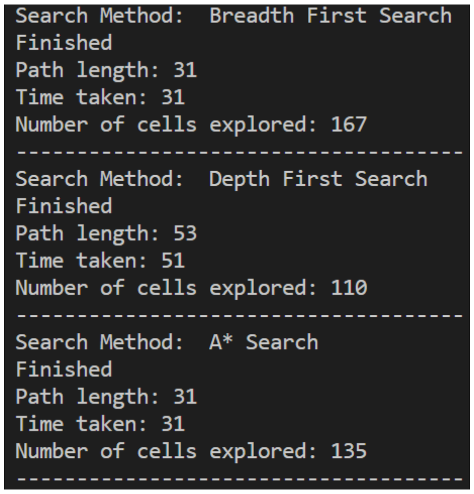
The data shows as the density increases the probability that it will be solved decreases. The probability comes to 0% after 40% density. The data also shows that A* was the most efficient algorithm for finding the target in the grid. It had the smallest amount of average cells explored while BFS explored 141 cells and DFS explored 93 cells. The lengths of the algorithms varied. BFS had a length of 29, DFS had a length of 45 and A* had a length of 29. This shows that it took A* and BFS usually have a shorter amount of time and to find the target than DFS.

Reflection

To increase the speed of processing, a heap data structure was implemented in the project. The algorithm can visit fewer cells by using a priority queue, and a priority queue with a heap was built specifically for the A* search method. This way the A* method is more efficient then the other methods because it doesn't need to visit as many cells to find the best path.

Extensions





```
Search Method:  Breadth First Search
Finished
Path length: 31
Time taken: 31
Number of cells explored: 167
-------------------------------------
Search Method:  Depth First Search
Finished
Path length: 53
Time taken: 51
Number of cells explored: 110
-------------------------------------
Search Method:  A* Search
Finished
Path length: 31
Time taken: 31
Number of cells explored: 135
-------------------------------------
```

Here are some examples of my extension. In my extension I created two more cell types that include a web (which slows you down) and a boost (which speeds you up). The time taken to go along a regular cell is one, for a web it is two, and for a boost it is zero. Therefore sometimes the time taken to get to the end can be less than the actual number of tiles that make up the path. By building upon the given cell types, and adding a few methods to calculate the total time of the path based on each cell, I was able to make this program works successfully. As seen above The time taken for the DFS is 51 while the actual path is 53. In the maze displays(BFS, DFS, A* in order) you can see the white cells which signify the webs, and the cyan cells which signify the boosts. I made a random number of webs spawn in (between 5-9) and a random number of boosts spawn in (between 3-7). The green path sometimes covers these tiles but they can be better seen if a delay is added when calling the search algorithm in the SimulateMazeExtension file. All files used for the extension have the label "Extension" at the end of the name.