Maze Search Algorithms with Heaps

Abstract

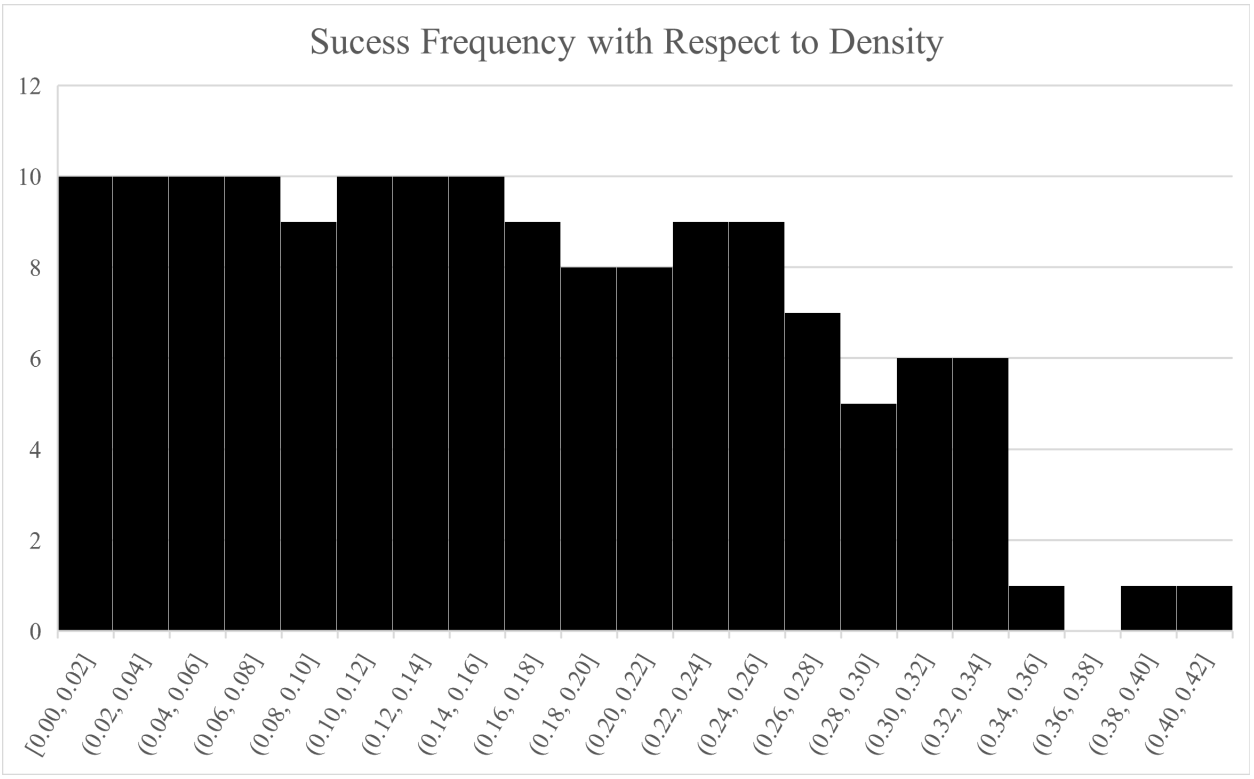
In this project we evaluate the performance of three maze search algorithms: Depth-First Search (DFS), Breadth-First Search (BFS), and A* Search. This project more than anything served to test our heap data structure. We tested the efficiency and effectiveness of the three algorithms considering a range of conditions. Each set of conditions was tackled by each of the three algorithms to ensure consistency. The analysis explores the relationship between the density of obstacles and the probability of reaching the target, the lengths of the paths found by each algorithm, and the average number of cells explored during the search. Experiments were conducted with varying obstacle densities from 0 to 1 to provide a comprehensive understanding of algorithm behavior.

Results

In the following table, we present the average success rate across five trials for the range of densities on the left. Samples were taken at a distribution corresponding with 0.01 density steps.

Algorithm / Density	DSF	BSF	A*
`[0.00,0.05)`	100%	100%	100%
`[0.05,0.10)`	96%	96%	96%
`[0.10,0.15)`	100%	100%	100%
`[0.15,0.20)`	88%	88%	88%
`[0.20,0.25)`	88%	88%	88%
`[0.25,0.30)`	64%	64%	64%
`[0.30,0.35)`	48%	48%	48%
`[0.35,0.40)`	8%	8%	8%
`[0.40,0.45)`	4%	4%	4%

Obviously, the success rate is the same across all algorithms as they use the same maze, however, this chart demonstrates the sharp drop in solvability between the `[0.30,0.35)` interval and the `[0.35,0.40)` interval. This is further evidenced in the following graph:



The following table represents the average length of a winning path produced by each algorithm over the same interval.

Algorithm / Density	DSF	BSF	A*
[0.00,0.05)	184.000	49.000	47.176
[0.05,0.10)	156.040	49.000	41.608
[0.10,0.15)	113.760	49.000	33.080
[0.15,0.20)	101.920	49.080	30.728
[0.20,0.25)	69.920	50.000	24.368
[0.25,0.30)	52.500	51.500	21.160
[0.30,0.35)	18.400	55.000	14.800
[0.35,0.40)	17.000	53.000	14.120
[0.40]	184.000	49.000	47.176

DFS consistently had the longest average paths and was very much luck based. Occasionally, especially in very restricted mazes it would have a very short path.

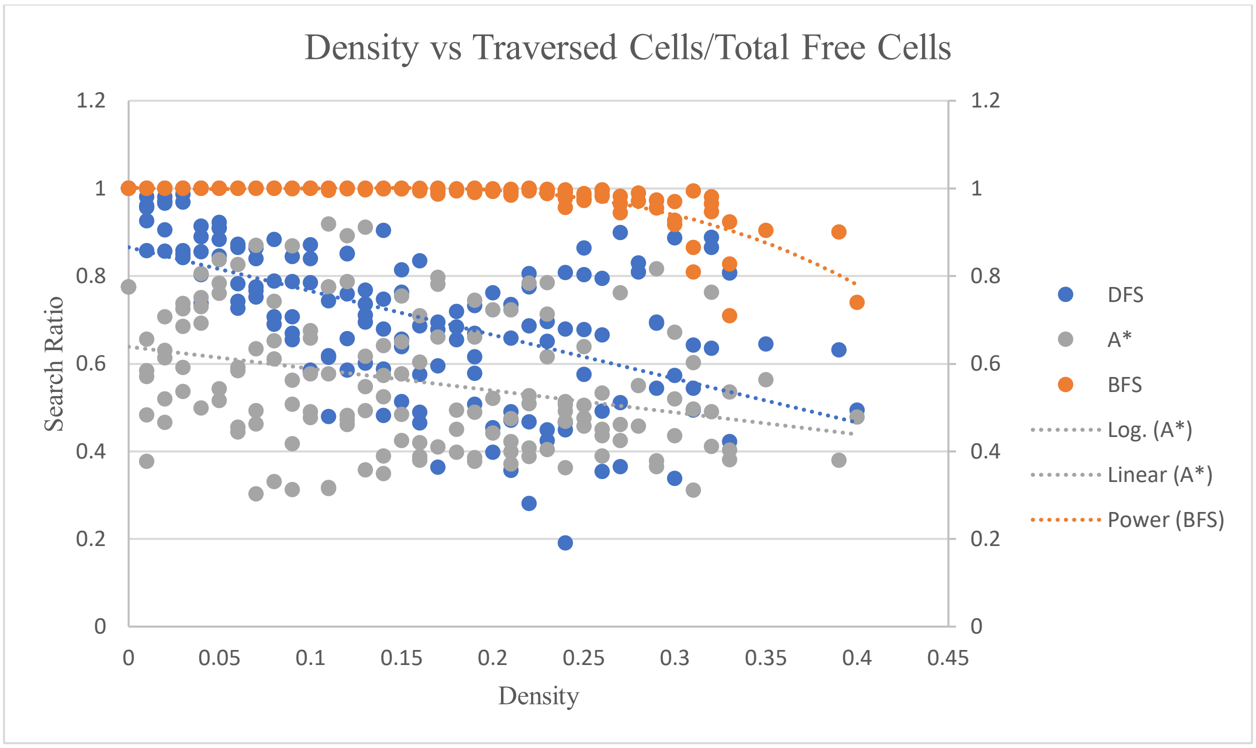
This following table represents the average number of free cells per maze across the five trials. As expected, these are invariant across algorithms as they share the same maze.

Algorithm / Density	DSF	BSF	A*
[0.00,0.05)	612.160	612.160	612.160
[0.05,0.10)	582.380	582.380	582.380
[0.10,0.15)	551.400	551.400	551.400
[0.15,0.20)	518.050	518.050	518.050
[0.20,0.25)	489.427	489.427	489.427
[0.25,0.30)	455.883	455.883	455.883
[0.30,0.35)	433.333	433.333	433.333
[0.35,0.40)	411.500	411.500	411.500
[0.40]	387.000	387.000	387.000

This final table represents the search ratio or the number of traversed cells over total free cells. This calculation is only done on winnable mazes as otherwise the data is useless as the algorithm will check all open cells for every algorithm.

Algorithm / Density	DSF	BSF	A*
[0.00,0.05)	92.232%	100.000%	64.893%
[0.05,0.10)	79.778%	99.993%	58.720%
[0.10,0.15)	69.987%	99.912%	57.067%
[0.15,0.20)	63.483%	99.700%	54.335%
[0.20,0.25)	55.077%	99.197%	53.168%
[0.25,0.30)	67.180%	97.574%	50.828%
[0.30,0.35)	65.812%	90.244%	50.122%
[0.35,0.40)	63.768%	90.151%	47.087%
[0.40]	49.354%	73.902%	47.804%

This table demonstrates the considerable advantage that the A* algorithm has in terms of efficiency. Something best evidenced by the following graph:



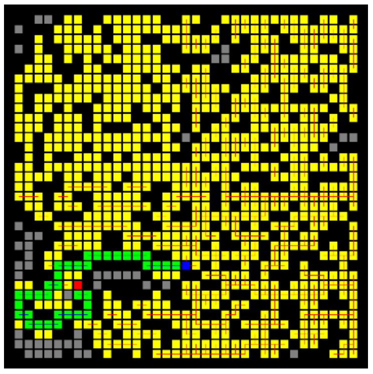
This graph demonstrates that while there is some degree of convergence of the trendlines near the upper bound of solvability for density in the maze, the A* algorithm far outperforms the other two algorithms in terms of efficiency. Furthermore, DFS outperforms BFS.

Now dealing with the more qualitative measures we have some other data.

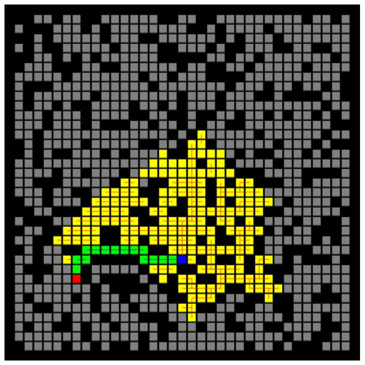
Breadth First Search under-performs in nearly every category except path length where it performs similarly to A*; however, A* still edges out a slight advantage.

Breadth first Search can occasionally out-perform DFS when the start and goal are very near each other.

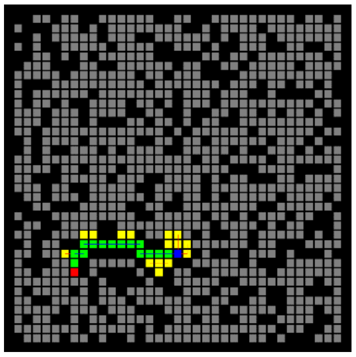
DFS:



BFS:



A*:



Depth First Search is overwhelmingly more variant in that its worst cases and best cases are nearly exactly as common as each other. If the DFS makes a wrong choice early on it will cycle through almost all possibilities. Breadth First Search is largely based on the individual characteristics of the maze as since it determines priority radially outwards impedances by obstacles Thus its somewhat more deterministic compared to DFS. A* is just exceptionally more suited.

Extensions

1. **Built In Data Extraction:**
   - The MazeSearchTestNoViz class is designed for data extraction outputting data as a csv file and printing out a table.
2. **Comprehensive Custom Setting Features:**
   - I implemented a comprehensive MazeSearchTest class file that allows for easy collection of visual data across algorithms and trials.
3. **Tweaking of A* Algorithm and other Methods**:
   - I made various small tweaks to a bunch of the algorithms here and there to make them better suited for my data collection and visualization needs. Including using the taxicab metric for the A* algorithm to allow for random start and end points.
4. **Hashed Heap with Binary Priority Traversal:**
   - For efficiency's sake my Heap Data Structure utilizes two hash maps, one gives a priority based on the node inputted, this is the priority map. On the other hand, the other map, the mirrorMap, is designed to take advantage of the contract between equals() and hashcode() to allow us to create a mode that only technically equals our desired node in the heap and then use its hash code to fetch the corresponding node in the heap. This is considerably more efficient compared to even binary traversal.
5. **Auto-Gif Conversion using Magick:**
   - I used the ProcessBuilder Class to automatically run command prompt and convert images into gifs.

Reflection

The project revealed insights into the strengths and weaknesses of each algorithm. Depth-First Search, while versatile, showed susceptibility to luck-based outcomes. Breadth-First Search underperformed in various categories, except path length where it matched A*. A* emerged as the algorithm of choice due to its superior efficiency.

The project's structured approach allowed for a deeper understanding of algorithm behavior, providing valuable insights into their applicability in different maze scenarios. The experience gained in implementing and analyzing these algorithms will undoubtedly prove beneficial in future algorithmic endeavors.