Searching Through Grids with Java

Abstract:

The real world problem I explored in this project was pathfinding and navigating through mazes. To better understand the problem in this project, I studied the search algorithms we used like DFS, BFS, and A*. The core CS concepts I used were algorithm design, object-oriented programming, and data structures like stacks, queues, and priority queues in this project. My key findings in this project were that if the density decreases, the probability increases and that A* also finds shorter and more efficient paths than DFS and BFS.

Results:

I ran experiments testing how well different sorting algorithms works like breadth first search, depth first search, and A*. I ran them with different difficulties (density) to see how they ran through each situation.

The metrics I am reporting are the probability of the target being reached based on each searching algorithm and the density of the obstacles.

Obstacle Density	DFS Probability	BFS Probability	A* Probability
0.1	0.9	0.95	0.98
0.2	0.85	0.92	0.96
0.3	0.75	0.85	0.91
0.4	0.65	0.78	0.85
0.5	0	0.	0.8
0.6	0	0.	0.75
0.7	0	0.	0
0.8	0	0.	0

0.9	0.	0.	0

Result 1: This table represents the probability of reaching the target in the maze for each searching algorithm, including BFS, DFS, and A*.

Question 1: There is an inverse relationship between the density and probability. The probability reaches 0 around an object density of 0.5.

Question 2: The relationship between the lengths of the paths depends on the density and the maze itself but BFS is usually shortest, DFS is the longest, and A* is the shortest and most efficient.

Question 3: The relationship between the average Cells and BFS, DFS, and A* is that DFS explores the fewest cells, then BFS, then A*.

Extensions:

I'm not doing an extension for this project.

Reflection:

1. Array-based heaps are easier to implement and they are more space-efficient because they can directly access every element and save memory. Node-based heaps are only better for when the heap is a tree.
2. If we implemented A* with a target distance of 0, it is breadth-first search. If we implemented A* with a target distance of infinity, it is Dijkstra's algorithm.

Acknowledgements:

I worked with Kamalani on the project. I also went to TA hours and got help from the TAs. I also used chatGPT to help me debug my code.