DV1619 Assignment 1

Tobias Gustafsson

September 20, 2024

1 Implementation

1.1 Breadth-first-seach

The Breadth-first-search algorithm was implemented using a queue. The starting node gets added to the queue. As long as the end node is not found and the queue is not empty, the program will look at the first element of the queue and add its neighboring nodes to the end of the queue. When a node is added to the queue, the program will store from which node it was reached.

1.2 Depth-first-search

The Depth-first-search was implemented recursively. The program begins with the starting node, and looks for adjacent nodes. If an adjacent node is found, the program will go there directly. If the end node is found, the recursion will return true and stop, otherwise it will try another path until all paths are explored.

1.3 A*

A-star was implemented using 4 dictionaries (hashmaps) that store data for each visited square: `g_scores` stores how far from the start a square is, `f_scores` stores f_score plus manhattan distance to the end, `root` from where the square was reached and `nodes` stores squares that are planned to be visited.

The algoritm begins with adding the starting square to the 4 dictionaries. Then, for as long as there are new squares to visit, the algorithm will do the following: Visit the square in `nodes` with the lowest f-score and add the values for its neighbors, that aren't already added, to `nodes`, `g_scores`, `f_scores` and `root`. The program finishes when no new nodes are available (no path found), or when the end squares i found (path found).

1.4 Comparison

All three algorithms found a path with the length 27 (including starting square). BFS will always find the shortest path, while A-star is likely to find the shortest path. DFS found the shortest path in this case, but DFS is not good for finding shortest path in general. For DFS the result also varies depending on in which order the program tries to look for a new node. In this program it first tried to go right, then down, left and up.

Looking at memory usage, DFS is the algorithm that uses the least amount of memory, it only stores the grid, and the path it currently on. The algorithm that uses the most memory is A-star. It uses 4 hashmaps to store data for each visited square. BFS uses a queue to keep track of which nodes to visit, which consumes memory. For larger grids, this way of storing all different paths could become an issue.

For maze problems, a-star is the most suitable for finding the shortest path in an efficient way.