ECM2423 CA Report

1.1

The maze solver can be seen as a search problem as we can take out given representation of the maze ‘#’ is a wall and ‘-‘ is a path and we can form a graph from this. This graph will consist of nodes which are the ‘-‘ path spaces, and the vertices represent nodes that are next to each other and can be traversed. Using this representation of the maze we can then carry out search algorithms to find a path between our designated start and end nodes.

1.2

A depth first search is a search algorithm that starts as a root and then will go as far as possible down a branch before it back tracking. It will then go back up the branch until it finds another branch that hasn’t been explored yet and will proceed to go down as far as possible. This repeats until either the desired node has been found within a branch or the entire graph has been explored.

My implementation of the depth first search can be found in the file dfs.py. To run the file for a specific maze, find the line at the bottom ‘depth\_first\_seach(“file\_name”)’ and change “file\_name” to the maze you would like to carry out the search on. The code assumes that the maze start is within the first line of maze text file and that the maze end is within the last line of the maze text file. It also assumes that there is one entrance or exit. If this is not the case the first instance of ‘-‘ in the first and last line will be taken as start and goal. The code will output the path which is represented as x and y coordinates where the x starts from 0 and represents characters horizontally and the y also starts from 0 and represents the characters vertically. Also outputted to the console is the length of the path , the number of unique nodes visited, and the time taken to execute the search. There will also be 2 files outputted after the search ,the first ‘maze-solution.txt’ which is already attached within the CA submission will show the path on the maze where ‘x’ represents the nodes in the path. The second file ‘Visited-nodes.txt’ outputs all the nodes visited onto the maze where the nodes visited are represented by x.

Depth First Search Results

|  |  |  |  |
| --- | --- | --- | --- |
| Maze Name | Time (in seconds) | Path Length | Unique Nodes Visited |
| Small | 0.04037833213806152 | 27 | 78 |
| Medium | 1.884493350982666 | 641 | 6,526 |
| Large | 124.70268940925598 | 1050 | 71,838 |
| VLarge | 365.2249128818512 | 4085 | 123,291 |

The depth first search results show that if the maze size is small then the search is very quick at finding a solution even though it may not be the shortest path. The number of nodes visited is only 2.89x larger than the path length. As the maze size increases the time taken increases linearly ,this is because depth first search has a linear time complexity. This makes it still quick on the VLarge maze taking into account the increase in the maze size. However, as the size increases the number of unique nodes visited compared to the path length increases exponentially. For Medium it is 10.18x , for large it is 68.42x and for VLarge it is x30.18. The VLarge maze may be an anomaly because based on which direction of node are explored first the depth first search may find a faster solution. Overall, the depth first search in terms of time is efficient at finding a solution but it may not be finding the most efficient path.

1.3

The other algorithm I have chosen to solve the mazes is a breadth first search. More specifically a bidirectional breadth first search. A breadth first search is a search algorithm where starting from a root it explores all the nodes on the same depth before moving onto the next depth. This happens until the goal is found or the entire graph has been explored. The bidirectional breadth first search means that we start a breadth first search from both the start and the end. When the start and end search meet, we have our solution by combining both the paths together.

Originally, I intended to just carry out a regular breadth first search but as you can see from the results table below because of the amount of overhead required when carrying out the search the time complexity is exponential so when it came to search the VLarge maze it didn’t complete within twenty minutes.

Breadth First Search Results

|  |  |  |  |
| --- | --- | --- | --- |
| Maze Name | Time (in seconds) | Path Length | Unique Nodes Visited |
| Small | 0.01998114585876465 | 27 | 75 |
| Medium | 1.7709846496582031 | 321 | 8,446 |
| Large | 100.2120418548584 | 974 | 81,969 |
| VLarge | N/A (after 20 minutes) | N/A | N/A |

I choose a bidirectional breadth first search as I wanted to use an algorithm that would find the most optimal path through the mazes and would have a lower time for completion than the depth first search.

To run the implementation of the algorithm it is the same as the depth first search where you change ‘breadth\_first\_search(“file name”)’ to run the search and everything will be outputted to the console and the 2 paths and explored nodes files will be updated.

Bidirectional Breadth First Search

|  |  |  |  |
| --- | --- | --- | --- |
| Maze Name | Time (in seconds) | Path Length | Unique Nodes Visited |
| Small | 0.0039980411529541016 | 27 | 66 |
| Medium | 0.29656171798706055 | 321 | 3,630 |
| Large | 13.044541358947754 | 974 | 30,597 |
| VLarge | 772.0746405124664 | 3692 | 190,609 |

The results from the bidirectional breadth first search show that it is very quick for smaller mazes compared to the depth first search, the small maze is roughly 10.1x quicker than the depth first search. The medium is 6.35x and the large maze is 9.56x quicker. The medium maze anomaly may be again because of preferences of direction of nodes. This shows that the bidirectional breadth first search is significantly quicker than the depth first search and the normal breadth first search. However, the VLarge is 0.47x slower than the depth first search this is probably because the amount of overhead required to store all of the possible paths becomes very large when working with that size of maze. If the maze size was to then be increased again the algorithm may not be able to find a solution in a reasonable amount of time. The bidirectional breadth search first does however show improvement as it was able to complete the VLarge maze is under 20 minutes. The path length is smaller for all of the mazes with the bi directional breadth first search over the depth first search which means that one of the goals with this second algorithm has been achieved. Interestingly the number of nodes visited varies for each maze compared to the depth first search. Again, this is probably down to directional preference when adding paths to the queue and the solution of the maze. Overall, the bidirectional breadth first search is superior to the depth first search in almost all aspects apart from time on very large mazes.