Motion Planning (RBE550)
Programming Assignment 1: BFS and DFS Algorithm Implementation

Assignment By:
Swapneel Dhananjay Wagholikar (WPI ID: 257598983)

1. Breadth First Search (BFS) Algorithm:

Explanation:
In BFS Algorithm, queue data structure is used which works on the principle of first in first out (FIFO). In the exploration of the grid, the algorithm starts with a start_node which is given as an input. Then it starts exploring its neighbors and so on. In doing so, it continues to append nodes into the queue and pops it out in order of its append.

Step 1: the start node is visited.
Step 2: the neighbors are explored in the order specified and pushed into the queue (level 1).
Step3: then explore the right node by pushing its neighbors into the queue (level 2).
Step4: However, these neighbors (level 2) are now at the end of the queue hence the other neighbors of the start node (level 1) are explored first.

Therefore, this algorithm is referred to as the Breadth First Search Algorithm as it explores as per FIFO (queue data structure).

Pseudocode:
1. Initialize start node, goal node, a queue and a “visited” matrix with 0 values
2. Set visited value of start node as 1 and push it in the queue
3. Iterate while length of queue is greater than 0
    a. Pop the first object (u) in the queue (increment step count for each visited node)
    b. Check if object is the goal node (break condition)
    c. Explore the neighbors (for loop)
        i. create a node (v) and specify the parent of v as u
        ii. Push the v node into the queue
        iii. If you have already visited, then remove the node from queue
        Iv. set visited = 1 for v
4. Use the generate_path function to return the path from start node to the goal node

2. Depth First Search (DFS) Algorithm:

Explanation:
In DFS Algorithm, stack data structure is used which works on the principle of last in first out or first in last out. In the exploration of the grid, the algorithm uses its recursive step to go into the depth of the branch till it reaches the end.

Step1: the start node is visited.
Step2: the neighbors are explored in the order specified.
Step3: the algorithm enters the node and then explores the neighbors of the current node.
Step4: when it finds suitable neighbors, it enters the same function again in a recursive manner.

Therefore, it goes on entering the function till it reaches a goal node.

Pseudocode:

1. Initialize start node, goal node, a stack and a “visited” matrix with 0 values
2. Step recursive function with input as the start node
    a. Set visited as 1 for input node
    b. Check if input node is goal node, if yes, set found = 1 and return
    c. Explore neighbors (for loop)
        i. If visited, continue to other neighbors
        ii. Recurse into the function using the neighbor node as input
        iii. If found = 1, return
    d. Return if no new neighbors found (reached end of the branch)
3. Use the generate_path function to return the path from start node to the goal node

Differences between BFS and DFS:

1. BFS works by exploring the breadth of the tree first, and then goes deeper unlike the DFS approach where the algorithm works by exploring the depth of the tree first, and then explores other branches.
2. BFS works on the First In Last Out type data structure (queue) whereas DFS works on the Last in First Out type data structure (stack).
3. BFS gives the shortest path whereas DFS may traverse through a lot of edges before reaching the goal node.

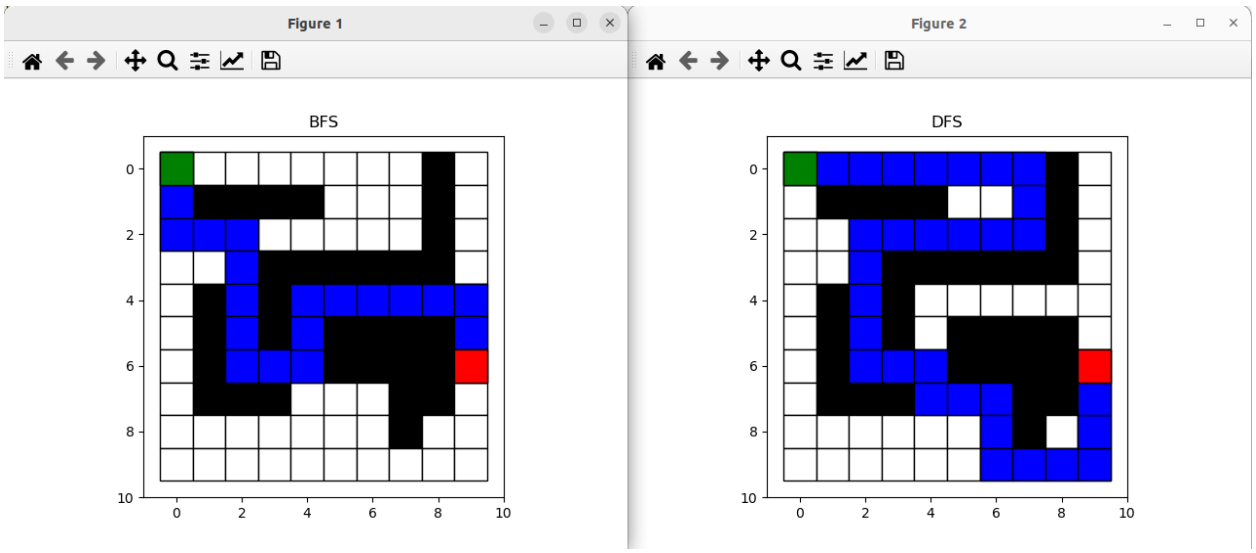
Similarities between BFS and DFS:

1. BFS and DFS are both exploratory techniques.
2. Both algorithms ignore the "cost to come" from the start node to the current node or the heuristics to the goal node.

Test Examples, Test Results and Explanation:

1. **For given test case (map.csv) in the assignment:**

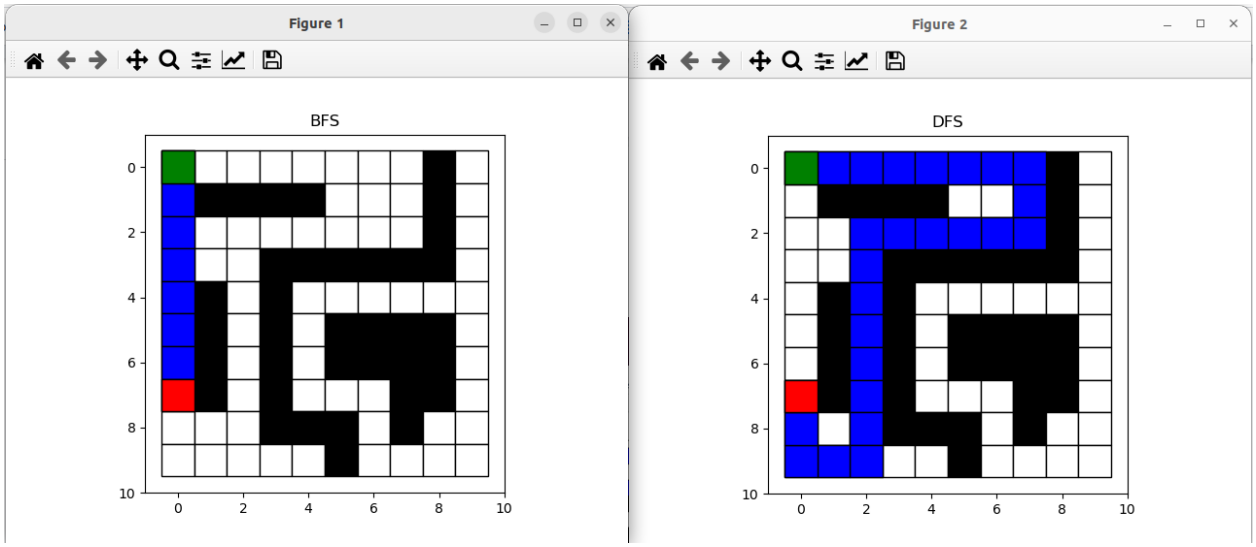
```
(base) swapneel@swapneel:~/rbe550/BFS, DFS - Swapneel$ python main.py
It takes 64 steps to find a path using BFS
It takes 33 steps to find a path using DFS
```



There is a difference in path traveled by both these algorithms. BFS explores the grid as per queue and finally follows the path by reversing as per the parent node while We see that the DFS goes on exploring the right node as the algorithm works on depth exploration.

2. **User Test Case 1:**

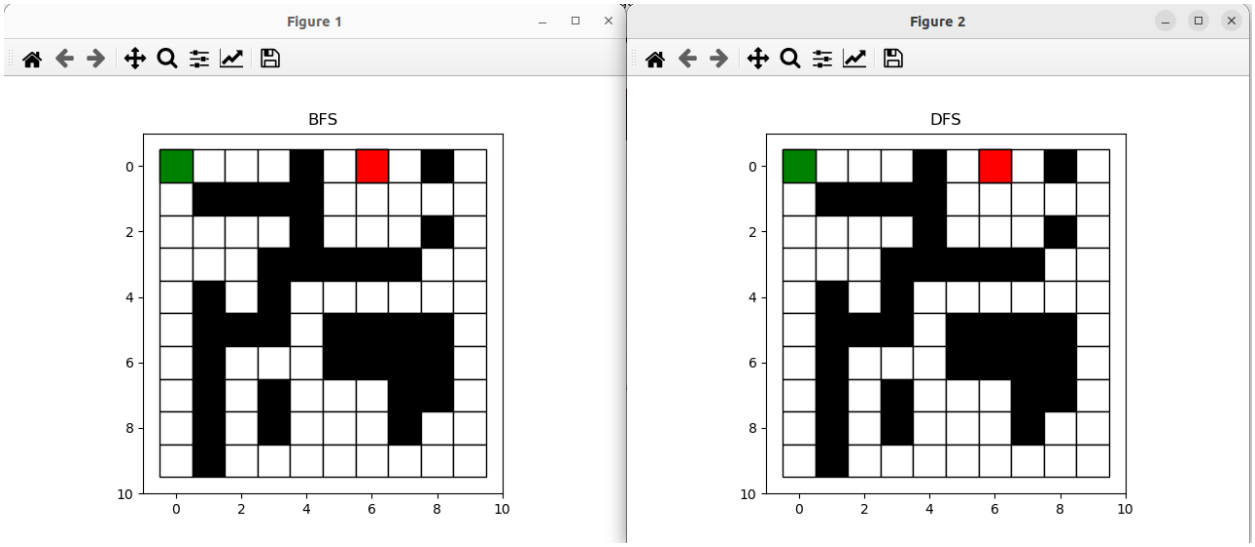
```
(base) swapneel@swapneel:~/rbe550/BFS, DFS - Swapneel$ python main.py
It takes 26 steps to find a path using BFS
It takes 29 steps to find a path using DFS
```



In this test example, DFS clearly takes the longest route as the algorithm explores the right node first every time. But, BFS finds the shortest path.

3. **User Test Case 2:**

```
(base) swapneel@swapneel:~/rbe550/BFS, DFS - Swapneel$ python main.py
No path found
No path found
```



In this test case, both the algorithms fail to find the path as the path is completely blocked by obstacles.

References:

1. https://favtutor.com/blogs/breadth-first-search-python
2. https://medium.com/nothingaholic/depth-first-search-vs-breadth-first-search-in-python-81521caa8f44
3. https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/