CS472 Assignment 2

1) Describe **Breadth First Search (BFS) algorithm** and **Depth First Search (DFS) algorithm** with examples, describe their complexities and differences.

2) Implement **Breadth First Search (BFS) algorithm** and **Depth First Search (DFS) algorithm using Python. Includes output taking at least two examples.**

Due Date: 14th September, 2023

**** Code for question 2 in hw2-2.py ****

Question 1:

Breadth First Search and Depth First Search are the two most popular methods of traversing graphs and tree structures and searching for specific nodes or vertices within the graph / tree data structure. “Breadth First” refers to the method of starting at the root/”source” node, and then descending the graph/tree structure one level at a time, moving on after each node in the current level has been visited. If you were to imagine a tree data structure as a pyramid, Breadth First Search would work by exploring every room in the top level of the pyramid, then going down a level, exploring all the rooms on that level, and so on until all rooms within the pyramid have been explored. In contrast, “Depth First” refers to the method of starting at the root/”source” node, choosing its first child node, navigating to it, then repeating this process recursively until you hit a leaf node, at which point you will visit the next child of the closest parent node and start the process all over again until all nodes have been visited. If you were to imagine a tree data structure as a pyramid, Breadth First Search would work by exploring the room at the very top of the pyramid, then the the first room you find on the next floor, followed by the first room you find on the next floor, until you reach the bottom, at which point you go up one floor and start the process again with the next room you find.

As far as complexities go, BFS and DFS have the same big-O complexities in both space and time measurements. Their space complexity is O(V) for the queue data structure / recursive call stack used in the algorithms, where V is the number of vertices (nodes). Their time complexity is O(V+E) where V is the number of vertices (nodes) and E is the number of edges in the graph/tree data structure. Although their Big-O space complexity is the same, DFS typically requires less memory, assuming it is implemented recursively. It is also worth noting that only BFS can guarantee completeness, meaning it will always find the shortest in an unweighted graph, assuming one exists, whereas DFS can not guarantee this. BFS is typically implemented using a queue, whereas DFS is typically implemented using a stack.

Question 2 Code Output:

1.
```
nayatrov@ariahas-MacBook-Pro ai-hw-2 % python3 hw2-2.py
BFS traversal (starting from vertex 2):
[2, 0, 3, 1]
DFS traversal (starting from vertex 2):
[2, 0, 1, 3]

BFS traversal (starting from vertex 'A'):
['A', 'B', 'C', 'D', 'E', 'F']

DFS traversal (starting from vertex 'A'):
['A', 'B', 'D', 'E', 'C', 'F']
```