# Unit 6 Activity: Graph Theory and Graph Algorithms

1) Breadth-First Search (BFS) is one of the most basic graph algorithms, and forms the basis of a number of more advanced topics within graph theory. The main goal of BFS is to take a graph, denoted by *G = (V, E),* where *V* represents the vertices in the graph and *E* represents the edges, and find the shortest path between the source vertex (analogous to the root node in the tree data structure) and all other vertices that are reachable from that source vertex. In this case, the shortest path is simply denoted as the number of edges traveled from the source vertex to the destination vertex. In doing so, BFS produces what is referred to as a *breadth-first tree*, where the source vertex is set as the root node and all reachable nodes are set as children of that root.

As indicated by its name, BFS works in a breadth-first manner, meaning that all reachable vertices at a specified distance, *k,* are discovered before all vertices that are reachable at *k* + 1 are discovered. Algorithmically, this is done by "coloring" the vertices as they are visited. An unvisited vertex is denoted as a "white" vertex, vertices that have had all edges extended from them explored are marked as "black," and vertices that have been visited, but whose edges have not all been explored, are denoted as "gray." Therefore, a BFS algorithm allows for each reachable vertex in the graph to be explored, and the path between the source vertex and each reachable vertex to be tracked.

BFS is an efficient algorithm, with a total running time of *O(V + E)*. It also paves the way for more useful graph-theory principles.

Implement the BFS algorithm, and test it using some sample graph examples. You can implement graphs using the data structure of your choice.

2) Depth-First Search (DFS) is similar to BFS. While the goals of each algorithm are the same, the ways we reach them are very different. As implied by its name, DFS searches as "deep" into a graph as possible before pulling back and exploring other paths. For example, if the source vertex has two adjacent edges, DFS will pick one and explore it fully before backtracking and moving on to the second.

DFS is also aided algorithmically by "coloring" each visited vertex, in the same manner as BFS. However, a vertex is not marked as being "black" until its adjacency list has been completely discovered and explored, which in turn can only happen when each vertex in that adjacency list has also had its adjacency list fully explored. Whereas BFS explores each current adjacency list at the same level, DFS picks one and explores it fully before moving on to the next one.

DFS has many attractive characteristics, such as the *parenthetical structure property* and the *white-path theorem*, each of which is useful when other algorithms attempt to extend the DFS algorithm. Similar to BFS, the running time of DFS is denoted by *O(V + E)*.

Implement the DFS algorithm and test it using some sample graph examples.  You can implement graphs using the data structure of your choice.