Lab 11

Data Structures

Gabe Imes, Joe Schnizer, Eric Bridgens

The objective of this lab was to learn about creating and traversing graphs. This concept is important for a career in CS because graphs are a good structure to use when dealing with networks where a relationship between one data point and another data point is important information such as a map, or a social media application. We implemented the depth first by traversing through the graph and checking if a certain node has been visited. If the node has been visited, we call the function recursively using a neighboring unvisited node. For breadth first search we mark each node false for a Boolean that checks if the node has been visited. We then add an unvisited node to a queue and go through each possible path adding the nodes along the path to the queue as well. As for choices of data structure, we selected a queue for BFS as it allows us to enqueue and dequeue visited nodes in the correct order and then output them in that same order as they are dequeued. Our DFS method uses recursion, so it does not rely on a stack or queue to function, but instead outputs the visited node each time it recurses through the function.

BFS and DFS have the same time complexity. In terms of memory DFS requires less memory than BFS because DFS starts a different path from each node rather than restarting from the beginning each time it is finding a new path. If the nodes that you are searching paths for are closer to each other, then a BFS would be the better search method to use. However, if the graph has a large amount of data, then a DFS would be the better search method to use.











