Joey Troyer

Vishwanathan Roopa

CS 372

Lab 4

03 April 2023

INTRODUCTION

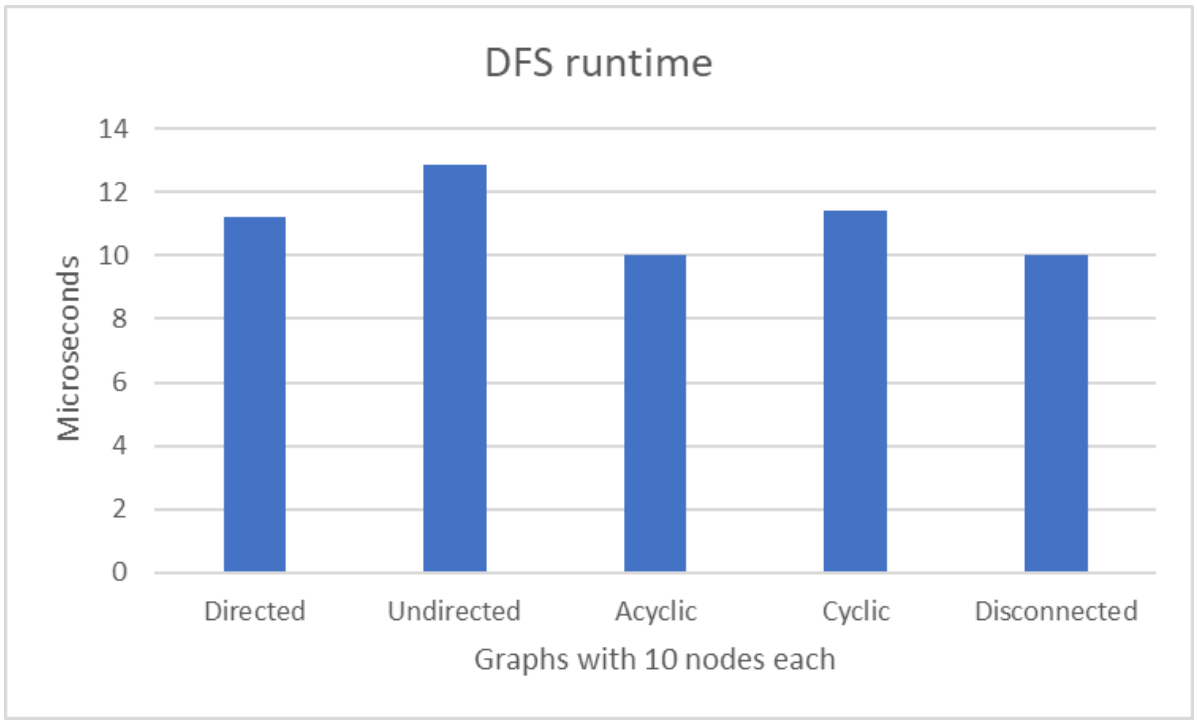
In this lab, we were tasked with implementing a graph using an adjacency list and implementing h a depth first search (DFS) algorithm to traverse the graph. We used DFS to find out in which order we were visiting each node. The motivation for this lab was to understand how graphs are implemented and how they could be used. It was also to learn how to implement the DFS algorithm commonly used to traverse graphs. The problem we had to solve was how to create a working graph by reading in a list of nodes separated by a tab from a text file. We then used the DFS algorithm to traverse a graph and keep track of the order in which nodes were visited each node had a previsit and a post-visit attribute to it so each time we visited a node we would save its place in either the previsit or postvisit. Using DFS we were able to see in what order all the nodes in the graph were visited.

METHODS

The method I used to develop my graph and DFS are based of the code supplied in the instructions and the book. I created the node class to represent individual nodes in the graph.

Each node has the attributes name, id, pre-visit, and post-visit. I also created a graph class that represents the actual graph and manages the nodes and their edges. The graph class includes a list of all the nodes and an adjacency list to store all the edges of each node. It also includes a scan method that reads a tab-separated text file containing nodes and their edges. For each set of nodes, the program will add the node only if it has not already been added and then it will connect the edge to the proper nodes. I implemented a recursive DFS function to traverse the graph. The function takes a node and the graph object, and then it traverses the graph, updating the previsit and postvisit values for each node accordingly. Inside the testDFS function, it performs DFS on every node in the graph that that has not already been visited so that it can also set the previst and postvisit of disconnected nodes.

RESULTS



The time it takes to run DFS seems to vary by about 10 microseconds. I believe this to be because of any other processes running on the pc are slowing the algorithm down. However, the average run time appears to be consistent with the expected time complexity of $O(|V| + |E|)$.

DISCUSSION

In this lab, we successfully implemented a graph using an adjacency list and implemented a depth-first search (DFS) algorithm to traverse the graph. Which was able to track all the connected nodes to the graph aswell as disconnected nodes. Some issues with this implementation could be that when handling very large and interconnected graphs, an adjacency matrix might perform better. Another issue is that the input file must be correctly formatted for it to work. We could add some error handling to make it easier to use and less prone to breaking.