Programming Assignment 2 Report – Abigail Miller

Output Table from code:

M§M BFS DFS  
MM§MNode 1 Node 2 Distance Time(ms) Distance Time(ms)  
MM§MN\_0 N\_1 1 1 1 0  
MM§MN\_0 N\_2 3 0 23 0  
MM§MN\_0 N\_3 6 0 24 0  
MM§MN\_0 N\_4 24 0 17 0  
MM§MN\_0 N\_5 6 0 17 0  
MM§MN\_0 N\_6 4 1 3 0  
MM§MN\_0 N\_7 7 0 5 0  
MM§MN\_0 N\_8 19 0 15 0  
MM§MN\_0 N\_9 22 0 16 0  
MM§MN\_0 N\_10 8 0 18 0  
MM§MN\_0 N\_11 13 0 20 0  
MM§MN\_0 N\_12 10 0 6 0  
MM§MN\_0 N\_13 14 0 10 0  
MM§MN\_0 N\_14 20 0 15 0  
MM§MN\_0 N\_15 12 0 21 0  
MM§MN\_0 N\_16 19 0 21 0  
MM§MN\_0 N\_17 17 0 8 0  
MM§MN\_0 N\_18 20 0 13 0  
MM§MN\_0 N\_19 23 0 14 1  
MM§MN\_0 N\_20 18 0 22 0  
MM§MN\_0 N\_21 22 0 23 0  
MM§MN\_0 N\_22 21 0 9 0  
MM§MN\_0 N\_23 24 0 10 0  
MM§MN\_0 N\_24 24 0 15 0

For this project, I decided to use an adjacency list to represent my graph as opposed to an adjacency matrix. I chose this because the given graph has relatively few connections between nodes and has many nodes. As such, using a matrix would take up considerably more memory compared to an adjacency list. Additionally, the text file being in a format of adjacent nodes was easier to implement into a list using the file scanner.

If I had a node at a shallow depth, I would use BFS. This is because DFS could go the wrong direction and won’t find the shallow node until it has completely traversed the depth of the tree. Meanwhile, BFS would find it in the first or second iteration. However, if I had a node at a deep depth, I would use DFS because it has a higher chance of finding the node faster than BFS.

The time in Milliseconds was too quick to capture, so here is the chart using nanoseconds

MM§M BFS DFS  
MM§MNode 1 Node 2 Distance Time(ms) Distance Time(ms)  
MM§MN\_0 N\_1 1 132800.0 1 0.0  
MM§MN\_0 N\_2 3 34900.0 23 0.0  
MM§MN\_0 N\_3 6 41600.0 24 0.0  
MM§MN\_0 N\_4 24 145900.0 17 0.0  
MM§MN\_0 N\_5 6 34900.0 17 0.0  
MM§MN\_0 N\_6 4 27500.0 3 0.0  
MM§MN\_0 N\_7 7 49400.0 5 0.0  
MM§MN\_0 N\_8 19 98600.0 15 0.0  
MM§MN\_0 N\_9 22 148500.0 16 0.0  
MM§MN\_0 N\_10 8 39100.0 18 0.0  
MM§MN\_0 N\_11 13 54300.0 20 0.0  
MM§MN\_0 N\_12 10 65700.0 6 0.0  
MM§MN\_0 N\_13 14 62000.0 10 0.0  
MM§MN\_0 N\_14 20 84600.0 15 0.0  
MM§MN\_0 N\_15 12 54500.0 21 0.0  
MM§MN\_0 N\_16 19 71500.0 21 0.0  
MM§MN\_0 N\_17 17 67900.0 8 1.0  
MM§MN\_0 N\_18 20 79800.0 13 0.0  
MM§MN\_0 N\_19 23 103400.0 14 0.0  
MM§MN\_0 N\_20 18 63600.0 22 0.0  
MM§MN\_0 N\_21 22 90700.0 23 0.0  
MM§MN\_0 N\_22 21 97300.0 9 0.0  
MM§MN\_0 N\_23 24 84000.0 10 0.0  
MM§MN\_0 N\_24 24 85300.0 15 0.0

However, even in nano seconds, the time it took for DFS to run was so small the computer had a hard time picking it up.

Given this information, I would only use DFS because it runs at pico seconds.