Programming Assignment 2 – Intro to Algorithms (COMP 3270) – Jewels Wolter

Results:

Node 1	Node 2	BFS Time	BFS Distance	DFS Time	DFS Distance
N_0	N_1	0.0019	2	0.0021	2
N_0	N_2	0.0078	3	0.0062	3
N_0	N_3	0.0119	5	0.0422	25
N_0	N_4	0.0400	24	0.0262	18
N_0	N_5	0.0129	6	0.0081	5
N_0	N_6	0.0107	4	0.0072	4
N_0	N_7	0.0138	7	0.0093	6
N_0	N_8	0.0248	16	0.0172	12
N_0	N_9	0.0319	21	0.0250	17
N_0	N_10	0.0160	8	0.0281	19
N_0	N_11	0.0191	11	0.0312	21
N_0	N_12	0.0169	9	0.0103	7
N_0	N_13	0.0210	12	0.0122	8
N_0	N_14	0.0260	17	0.0181	13
N_0	N_15	0.0179	10	0.0303	20
N_0	N_16	0.0238	15	0.0341	22
N_0	N_17	0.0210	13	0.0122	9
N_0	N_18	0.0269	18	0.0193	14
N_0	N_19	0.0348	22	0.0203	15
N_0	N_20	0.0229	14	0.0372	23
N_0	N_21	0.0300	20	0.0391	24
N_0	N_22	0.0288	19	0.0141	10
N_0	N_23	0.0370	23	0.0150	11
N_0	N_24	0.0420	25	0.0219	16

Descriptions:

Breadth First Search-

BFS traverses a graph by first looking at all the nodes that are directly connected to it—staying at the closest depth—and then moving onto nodes that are directly connected to those child nodes and so on.

Depth First Search-

DFS traverses a graph by exploring each possible path fully until the node does not have any neighbors then recursively backtracking to the closest previous node that has neighbors.

Questions:

1. *Suppose you want to find a path between nodes at a shallow depth to your start node. Would you use BFS or DFS?*

To find a path between nodes at a shallow depth, I would use BFS. Since BFS can find the paths to the nodes most closely connected to the root first, this would be best for one close to the root node. In my implementation, the data shows that nodes like N_1, N_2, N_3, and N_6 have the fastest time and visit the least amount of nodes before finding it, making BFS the more efficient algorithm to find nodes with a shallow depth.

2. *Suppose that the end node is at a very large depth from the start node. Would you use BFS or DFS?*

To find a path to a node with a very large depth, I would use DFS. Since DFS traverses the graph by exploring each possible path to its end node fully before backtracking, it would be more efficient to use DFS in cases where the end node that is at a large depth needs to be found. In my implementation, the data shows that nodes at a large depth like N_24, N_4, and N_23 actually are found to have quicker time and less visited nodes to reach them then nodes like N_3 which is at a shallow depth.