COMP 3270 PA2

Node 1	Node 2	BFS Distance	BFS Time(ms)	DFS Distance	DFS Time(ms)
N_0	N_1	1	0	1	0
N_0	N_2	3	0	23	0
N_0	N_3	4	0	24	0
N_0	N_4	23	0	15	0
N_0	N_5	5	0	16	0
N_0	N_6	3	0	2	0
N_0	N_7	6	0	3	0
N_0	N_8	15	0	13	0
N_0	N_9	20	0	14	0
N_0	N_10	7	0	17	0
N_0	N_11	9	0	21	0
N_0	N_12	8	0	4	0
N_0	N_13	11	0	8	0
N_0	N_14	16	0	12	0
N_0	N_15	10	0	18	0
N_0	N_16	13	0	22	0
N_0	N_17	12	0	5	0
N_0	N_18	17	0	9	0
N_0	N_19	21	0	10	0
N_0	N_20	14	0	19	0
N_0	N_21	19	0	20	0
N_0	N_22	18	0	6	0
N_0	N_23	22	0	7	0
N_0	N_24	24	0	11	0

I went with an adjacency list to represent the graph for this programming assignment. The graph we were given didn't have that many connections between the many nodes present. Therefore, the adjacency list is more suitable than using a matrix because a matrix would take up more memory. The test case file was also formatted as adjacent nodes, so it made it easy to implement to a list.

If there is a node with shallow depth, BFS would be more suitable to use. DFS on the other hand could go in the wrong direction and not find the node until it traversed the depth of the whole tree. BFS finds shallow depth node within the first or second iteration. DFS would be useful if the node was at a large depth because it has a higher chance of finding the node faster than BFS.