Akoh Atadoga

CS 5120 Project 2C

**Introduction**

Breadth First Search (BFS) is an algorithm that can be used to find the shortest part in a graph. Theoretically, the running time of the BFS algorithm is . In this project I do a practical analysis of the BFS algorithm.

**Method**

I used python for the following:

1. Generate random graphs from a given number of nodes and number of edges
2. Implement Breadth First Search (BFS) to find the minimum path from a start node to and end node. If no path exists, it returns None.

**Result**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number | Number of nodes | Number of edges | Start Node | End Node | Shortest Path | Time taken (milliseconds) |
| 1 | 30 | 60 | 8 | 27 | ['8', '24', ‘27'] | 0.02574920654296875 |
| 2 | 50 | 200 | 30 | 35 | ['30', '6', '21', '35'] | 0.10085105895996094 |
| 3 | 100 | 1000 | 15 | 81 | ['15', '29', '81'] | 0.06818771362304688 |
| 4 | 200 | 4000 | 132 | 179 | ['155', '158', '170'] | 0.20813941955566406 |

**Analysis**

As seen from the table above, as the number of nodes and edges increases, the time taken to go from a node to another node increases. For example, when the number of nodes is 30, edges is 60. It takes approximately 0.03 milliseconds to go through 3 nodes. However, when the number of nodes is 100, edges is 1000. It takes approximately 0.07 milliseconds to go through the same number of nodes in the queue. The same can also be seen when the number of nodes is 200 and edges is 4000. It takes 0.2 milliseconds to go through just 3 nodes.

**Conclusion**

The practical performance is the same as the theoretical performance. From the theoretical perspective, the running time of BFS algorithm is . Meaning that as the number of nodes and edges increases, the running time increases which is the same as the result I got. A graphical representation is shown in the graph above.