Task 9.1P

Name- Anneshu Nag
Student ID- 2210994760

Q1. Given a graph G = <V, E> what is to be the running time of the depth-first search algorithm, as a function of the number of nodes n = $|V|$ and edge m = $|E|$, if the input graph is represented by an adjacency matrix instead of an adjacency list?

<u>Ans:</u> **We know that DFS algorithm searches every nodes (V) and edge (E) of a graph. So the running time will be, T (n) = O (V + E).**

If the input graph is represented by an adjacency matrix instead then the running time will be, T (n) = O (V^2).

Q2. After starting your first programming position your colleague asks you for advice on which algorithm they should choose. While your colleague studied programming until SIT232, they did not do SIT221 because they thought it would be too hard, therefore they have no idea how to decide on which algorithm to use. Because you are generous you decide to offer some advice.

Their problem requires them to find the shortest path from where a user is currently located to each of the major tourist locations in the city. To solve this your colleague had googled for an algorithm and came across two: Dijkstra's algorithm, which will solve it in O (V + E log V), and Floyd's algorithm, which will take O (V^3), where $|E|$ <= $|V|^2$. While you recall studying Dijkstra's algorithm during your SIT221 unit you have never heard of Floyd's algorithm. Regardless because your colleague's Google search had also given the time complexity you feel equipped to provide advice.

Discuss what factors your colleague should consider when deciding between the two algorithms and in which circumstances, they should choose one over the other.

<u>Ans:</u> **Djikstra's algorithm is utilised for finding the shortest path from the source node to all the other nodes. Meanwhile, Floyd's algorithm finds the shortest path between all pairs of vertices. One uses greedy approach while the other uses dynamic approach.**

Point to note that Djikstra's algorithm can be modified by making the nearest node as a new source, the previous source as already visited node so that it will pretty much perform like Floyd's algorithm.

In this scenario, a user needs to find the fastest way to reach various important tourist destinations from their current location. This can be done using Dijkstra's algorithm, but in practice, visitors might start their journeys from different starting points, requiring multiple iterations of the algorithm to discover all possible routes.

If the tourists starting from different locations each time then the Dijkstra's algorithm can be a good choice but if the routes between tourist's places do not change frequently and all the paths can be stored in the application then Floyd's algorithm will become the right choice for the scenario.

Mathematically,

We can write the time complexity of Djikstra's algorithm = O (E log (V))

We know the time complexity of Floyd's algorithm = O (V^3)

For a complete graph i.e. when Djikstra's algorithm is to calculated the shortest distance between all the pair of nodes iteratively, E = (V * (V - 1) / 2)

Time complexity for such scenario = O (V (E log (V)))

Placing the value of 'E' in the complexity we get,

$$= (V (V * (V - 1)) / 2) * \log (V)$$

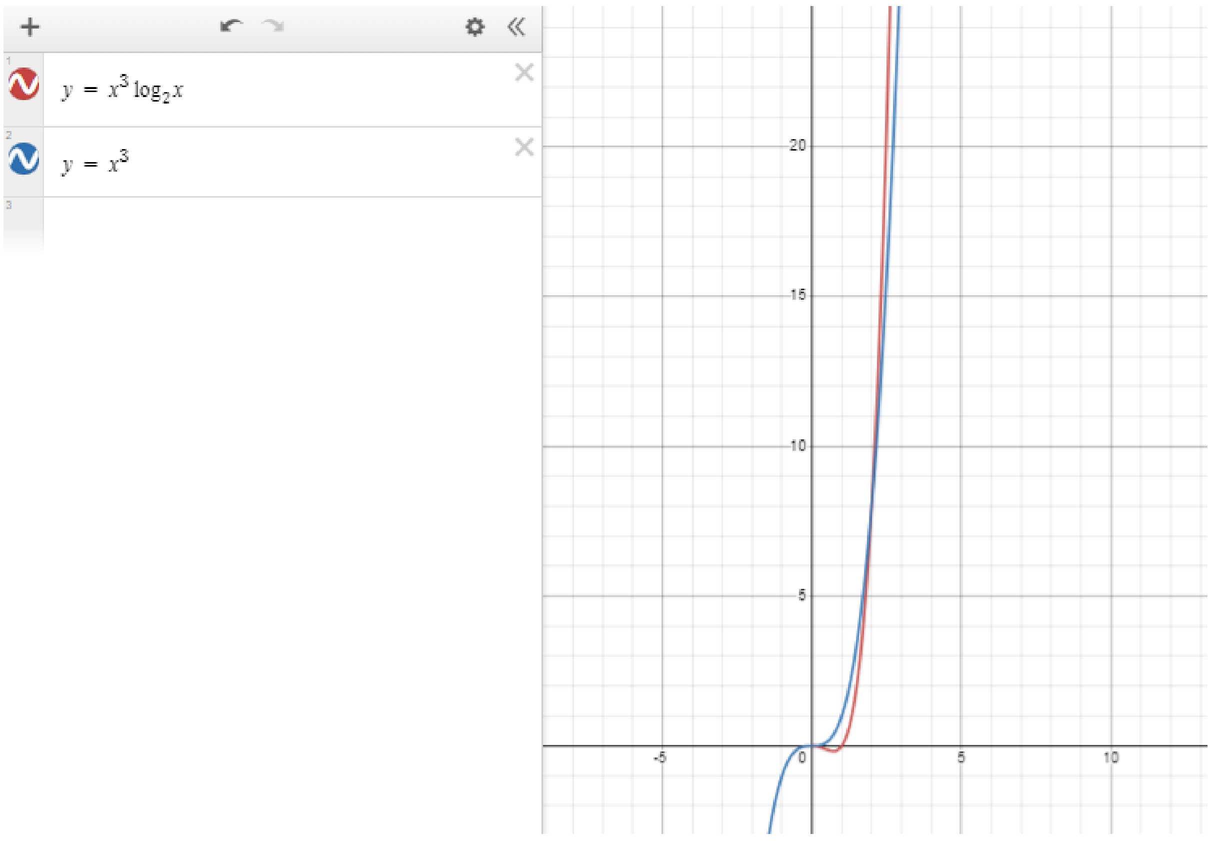
$$= V (V^2 - V) / 2 * \log (V)$$

$$= (V^3 – V^2) * \log (V)$$

$$= V^3 * \log (V) – V^2 * \log (V)$$

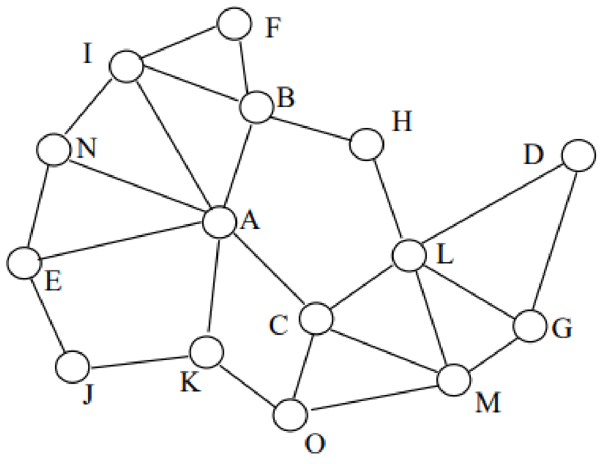
Hence, the complexity for the algorithm would be O (V^3 * log (V)) while Floyd's algorithm is O (V^3), which is faster. So we prefer Floyd's algorithm.

Graph:



Made using Desmos

Q3. Given the following graph draw both the Depth-first and Breadth-first search trees that will result if you start at Node A. *Note: when selecting between nodes you should use alphabetical order.*



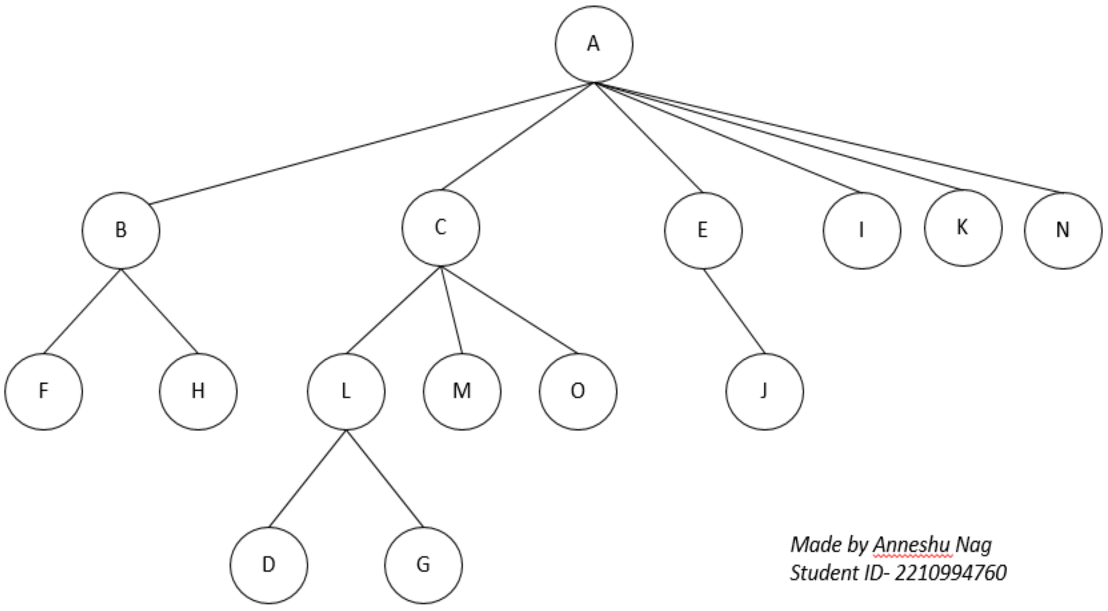
<u>Ans:</u> Depth-First Search:

A -> B -> F -> I -> N -> E -> J -> K -> O -> C -> L -> D -> G -> M

\

H *(Alternate path from L to H after backtracking)*

Breath-First Search:



Made by Anneshu Nag
Student ID- 2210994760