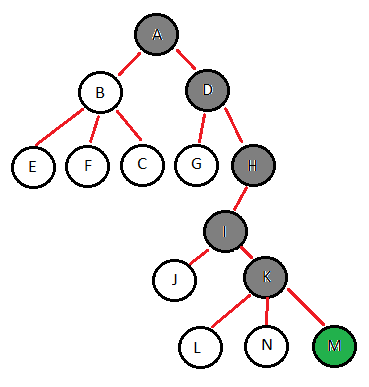
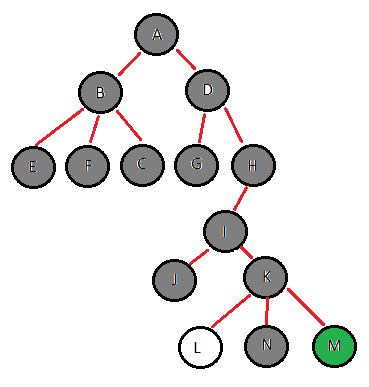
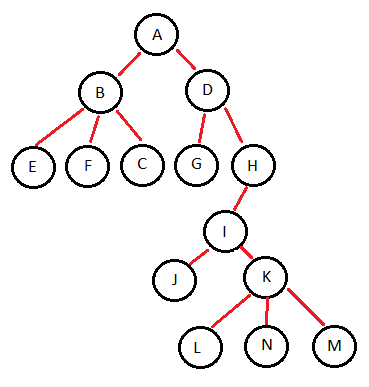
**Project 3: Maze Navigation**

In this project we demonstrated the power of informed search specifically in the context of maze navigation. A path through a maze can be portrayed as a tree of interconnected nodes (or a graph). We see that an informed search algorithm, particularly A\*, cuts the number of operations by a significant amount. A\*, which uses a heuristic to determine its next best path, proves to be an ideal algorithm for pathfinding both because it is *admissible* and *consistent.*

The source code for everything below can be found [here](https://github.com/Vardominator/CSULBProjects/tree/master/CECS451_ArtificialIntelligence/Project3_MazeNavigation/MazeNavigation/MazeNavigation).

**Sketch the search tree representing the maze:**



**BFS Search A\* Search**

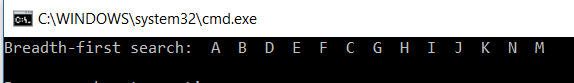
**What kind of search is implemented during the left (right) side traverse?**

Given that the robot has found all possible nodes (A through M), it has probably used a breadth-first search. Visually, BFS expands radially outwards and thus explores all possible states. This true only if it is not looking for a goal but merely searching.

**Describe and illustrate a greedy algorithm for finding the exit:**

I chose breadth-first search for my greedy algorithm. We can see that it visit nearly every node in the maze before finding M. Although complete, it is inefficient.

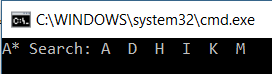




As demonstrated and drawn above, we see that BFS is guaranteed to find a solution if there is one. However, we might visit every possibility before doing so.

**Describe and Illustrate an A\* algorithm for finding the exit:**

****



This is a fairly typical implementation of A\*: a parent hash map keeping track of the path progression, a priority queue to explore the nodes with the best path first, and obviously a heuristic which is the Euclidean distance between nodes:

