Let’s do some complexity analysis

**The Breadth First Search algorithm**

Analyze the worst case running time complexity of this algorithm, by analyzing the cost of each line and the number of times it would repeat.

1: Set Discovered[s] = true and Discovered[v] = false for all other v
2: Initialize L[0] to consist of the single element s
3: Set the layer counter i = 0
4: Set the current BFS tree T = NIL
5: While L[i] is not empty
6: Initialize an empty list L[i + 1]
7: For each node u in L[i]
8: Consider each edge (u, v) incident to u
9: If Discovered[v] = false then
10: Set Discovered[v] = true
11: Add edge (u, v) to the tree T
12: Add v to the list L[i + 1]
13: Endif
14: Endfor
15: Increment the layer counter i by one
16: Endwhile

Now, let’s analyze it again together.

**The Depth First Search algorithm**

Can you follow the same method we just did in class to analyze the worst case running time complexity of the DFS algorithm?

1: Initialize S to be a stack with one element s
2: While S is not empty
3: Take a node u from S
4: If Explored[u] = false then
5: Set Explored[u] = true
6: For each edge (u, v) incident to u
7: Add v to the stack S
8: Endfor
9: Endif
10: Endwhile
Some proofs

Breadth First Trees

**Theorem.** Let $T$ be a breadth-first search tree, let $x$ and $y$ be nodes in $T$ belonging to layers $L_i$ and $L_j$ respectively, and let $(x,y)$ be an edge of $G$. Then $i$ and $j$ differ by at most 1.

**Proof.** Let $T$ be a breadth-first search tree, let $x$ and $y$ be nodes in $T$ belonging to layers $L_i$ and $L_j$ respectively, and let $(x, y)$ be an edge of $G$. Assume that $i$ and $j$ differ by more than 1.

Without loss of generality, assume that $j - i > 1$, with $x$ being discovered first. As the neighbors of $x$ are being explored, $y$ will be encountered and one of two things will happen:
1- $y$ will be added to layer $L_{i+1}$. Thus $j = i + 1$, which is a contradiction.
2- $y$ will not be added to the next layer because it is already discovered, which is a contradiction as well. Or if you want to add more details, this means that $y$ is in a layer $L_j$ such that $j \leq i$, which is a contradiction.

Depth First Trees

**Theorem.** Let $T$ be a depth-first search tree, let $x$ and $y$ be nodes in $T$, and let $(x, y)$ be an edge of $G$ that is not an edge of $T$. Then one of $x$ or $y$ is an ancestor of the other.

**Proof.** Suppose that $(x, y)$ is an edge of $G$ that is not an edge of $T$, and suppose without loss of generality that $x$ is reached first by the DFS algorithm. When the vertex $y$ is being examined as a neighbor of $x$, the only reason that the edge $(x, y)$ is not added to $T$ is that $y$ is marked “Explored.” Since $x$ was discovered first, then $y$ is discovered between the invocation and end of the recursive call DFS($x$), which means that $y$ is a descendant of $x$.