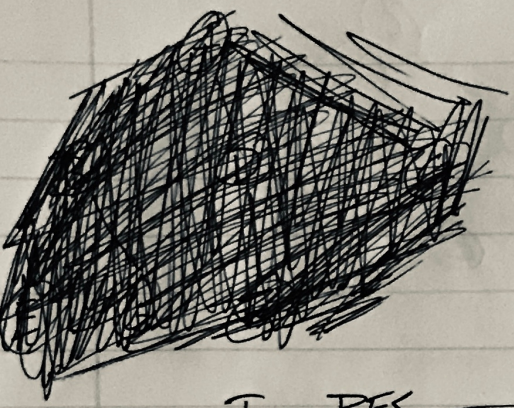
Christopher Patterson
EE 5301- Homework 1
10/12/18

Assuming I have full control over the BFS algorithm, I can keep two queue lists. One for generation X and one for generation X+1. This would allow me to search generation X in any order I want so any permutation of nodes at a given level is possible. Once I finish searching generation X, X+1 becomes X and new nodes found are added to a new X+1 generation.

Thus my assumption for problem 1+3 is that I can search a given level in any order (such as Lifo, FIFO, random, etc) regardless of how the nodes order in the queue are.

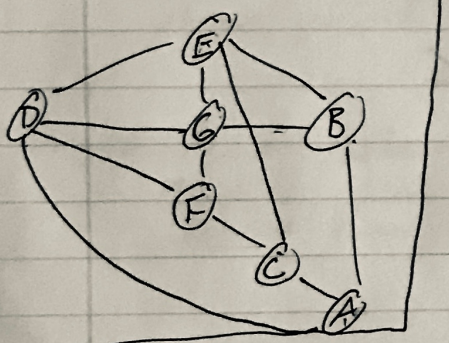
#1) I will draw a graph for each node as the root. Each node following will be put in their respective levels and every extraneous Path (say a node has 3 paths to the level above) will be omitted simplifying the visual representation.



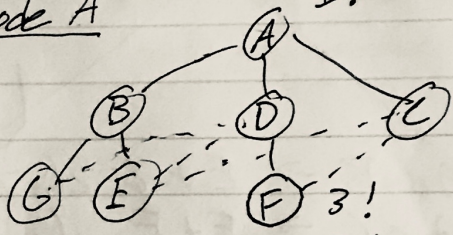
- A graph traversal is defined as I way to visit each node of a tree.
- At every level, there are $n_k!$ ways to visit all nodes if n_k is the number of nodes at a specific degree or level of the graph

In BFS →

Redraw graph

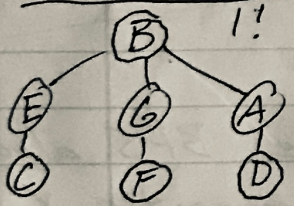


For node A



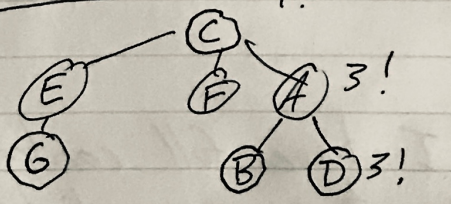
∴ 1! * 3! * 3! = [redacted] traversals from A

For node B



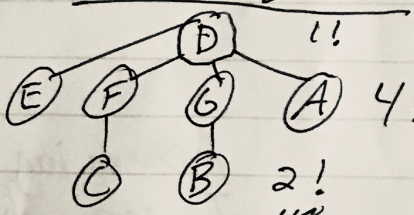
∴ [redacted] traversals from B

From C



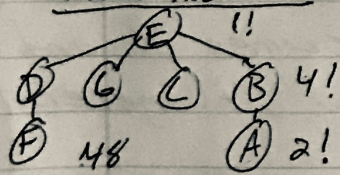
∴ [redacted] traversals from C

From node D



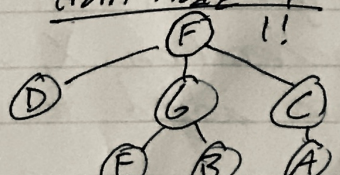
∴ 1! * 4! * 2! = [redacted] traversals

From node E



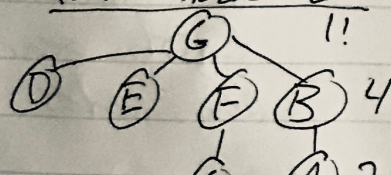
∴ [redacted] traversals from E

from node F



∴ [redacted] traversals from F

from node G



48

∴ [redacted] traversals

∴ The sum of traversals for each starting nodes graph is the total traversals:

3(48) + 4(36) = **288 total traversals of BFS**

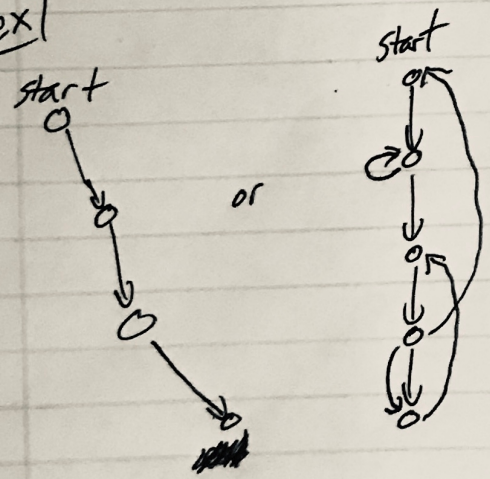
#2)

Directed

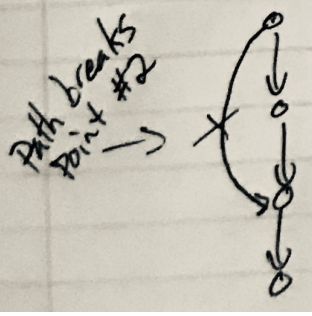
1. Graph should only have one forward path
2. No node should have more than 1 path out of it to a node which has not been visited yet
3. A node can have as many inputs & outputs as long as they follow the above point

→ ie: every node may have at most 1 child which has a larger degree than itself.

ex)



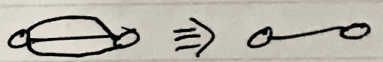
But not:



Undirected

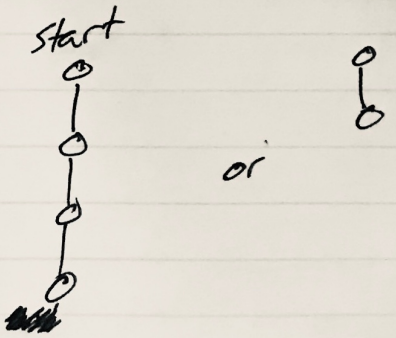
1. Each node may have a max of one input and one output path.
2. There may not be any cycles in the graph (ie. start connected to end unless its a graph of just 2 nodes)

*Note: I assume multiple paths from one node to another are simplified to one path ie

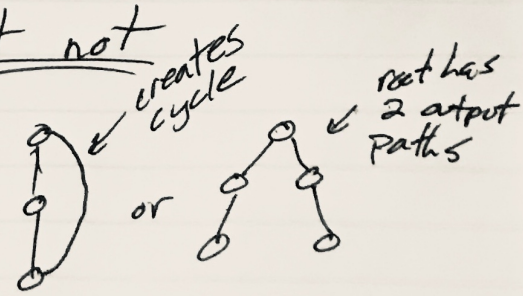


otherwise point 1 would be relaxed to say "all paths leaving node a V_i must connect directly to another node V_j."

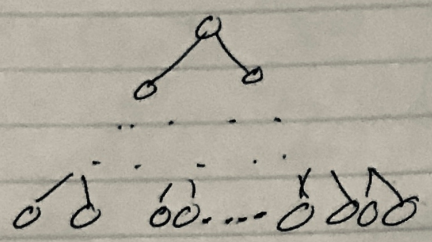
ex)



But not



#3) a) balanced: Assuming a balanced tree, every level has 2^L nodes with a max L defined as k

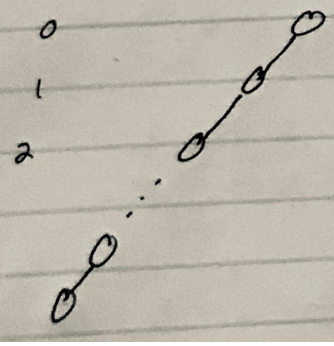


So each level can be traversed in any order for all 2^L elements. This means level 1 can be traversed 2! or 2 ways & level k can be traversed $2^k!$ ways.

∴ The total traversals for a balanced tree would be

$$\prod_{L=0}^{k} 2^L!$$

b) unbalanced: An extreme unbalanced tree would have all children nodes either to the left or the right of the parents. This means there is only one path from the root node to the leaf node.



∴ There is only one traversal.

There are less extreme unbalanced trees but each one would have different # of traversals.

#4) Prove its impossible to see more than 2 generations in the queue used by BFS.

Proof by contradiction

~~Assume there is a node V_i who is in the queue with a generation of $X+2$ a node from generation X~~

~~By definition of BFS, we will be~~

→ Assume nodes from generation X & $X+1$ are in the queue. *By definition of BFS, we search nodes in the queue youngest generation to oldest, thus we will be searching a generation X node next.

To see three generations or more in the queue, while searching generation X, we find a node of generation $X+n$ where $n>1$. Assume while searching node V_i of generation X, we find a node V_j of generation $X+n$. This means V_i is one edge from V_j so V_j has a generation of $X+1$.

This is a contradiction to the assumption that we would find V_j to be generation $X+n$, $n>1$

∴ It is impossible to see nodes from more than 2 generations in the queue of BFS.

*Note: When searching generation $X+1$, no generation X is left in the queue so we can denote $X+1$ as a new X and continue this proof.

$O(n^3)$

#5) a) $53 + 2n^3 + 4n^2 \log n \leq c \cdot n^3$ for some c & n_0

$53 \leq 53n^3 \quad n_0 = 1$
$2n^3 \leq 2n^3 \quad n_0 = 0$
$4n^2 \log n \leq 4n^3 \quad n_0 = 1$
} $53 + 2n^3 + 4n^2 \log n \leq 59n^3$

$\boxed{c = 59, \; n_0 = 1}$

not $O(n^2)$

- Assume $O(n^2)$

$\frac{c \cdot n^2}{n^2} \geq \frac{53 + 2n^3 + 4n^2 \log n}{n^2}$ find c & n_0

$c \geq \frac{53}{n^2} + 2n + 4 \log n$

↑ ↑

Unbounded, continually growing w/ n, thus c cannot be found

∴ not $O(n^2)$

$\boxed{O(n^3) \text{ is tightest}}$

b) $T(n) = 2T(n/2) + n^3$

$= 2[2T(n/4) + (\frac{n}{2})^3] + n^3 = 2^2 T(n/2^2) + \frac{n^3}{4} + \frac{n^3}{1}$ ← 4^0

$= 2^2 [2T(n/2^3) + (\frac{n}{4})^3] + \frac{n^3}{4^1} + \frac{n^3}{1}$ ← 4^0 $= 2^3 T(n/2^3) + \frac{n^3}{4^2} + \frac{n^3}{4^1} + \frac{n^3}{4^0}$

$= \sum_{k=0}^{\infty} \frac{n^3}{4^k} \rightarrow$ converges $= \frac{n^3}{3}$

Prove $O(n^3)$

$\frac{n^3}{3} \leq c \cdot n^3 \quad c = 1, \; n_0 = 0$

$\frac{0^3}{3} \leq 1 \cdot 0^3$ ✓

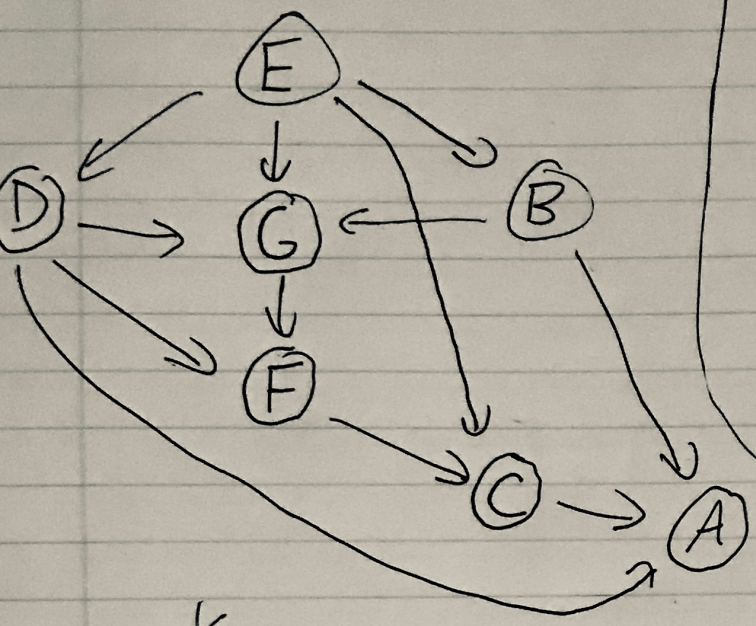
as n increases,

$\frac{n^3}{3} \leq n^3$ ✓

$\frac{1}{3} \leq 1$

∴ $O(n^3)$ Time complexity

#6) Redraw Graph



Info From the graph

- From This, we see every node is a dependent on (E) or a descendant of a dependent. ∴ E must be 1st
- A is a dependant of every node in some way. ∴ A must be last

Traversals

1) E → B → D → G → F → C → A

2) E → D → B → G → F → C → A

∴ There are only 2 traversals as dependencies are fairly linear from G → F → C → A

- B & D are the only two only dependant on (E) being chosen
- (G) must follow (B) + (D)
- (C) must follow (F)
- (A) must follow (B) (D) + (C)
- (F) must follow (D) + (G)