Andrew Taylor

Atayl136

Homework 6 – Graphs and Trees

1. How many ancestors does a node at level n in a binary tree have? Provide justification.

Given that if a tree's root is considered level 0, then any individual node has n ancestors equal to the level of the node n. Thus the root has no ancestors, which stands by definition. The definition of an ancestor is to trace back parent nodes, and only one parent is allowed per node, and we can only trace back one level at a time. Nodes cannot skip levels. Thus we come up with one ancestor node per level of descent - a node at level 2 would have 2 ancestors before it, and so on.

n = level, k = ancestors

at root n = k

for any child of node a level n, the level is n + 1

at the root k = 0

for the child of the root level = n + 1 and ancestors = k +1

for the grandchild the level = n + 2 and ancestors = k + 2

so for n = k at the root, n + 2 = k + 2 still at the grandchild

(k +2) -2 = (n +2) -2

= (k = n)

So the number of ancestors equals the level.

2. Prove that a strictly binary tree (regular binary tree) with n leaves contains 2n-1 nodes. Provide justification.

A binary tree with n leaves that is regular has 2 children per node by definition, except at the leaf. Such a tree would have n child nodes as leaves at the bottom layers. Taking a base case of one leaf we can prove the tree contains 2n-1 nodes by induction. Let's use a root with two children. Here it is true that at level 1 there are two leaves, as child nodes, and the root in the tree, thus 2n-1 = (2*2)-1 = 3. Adding a leaf and generalizing to k+1 leaves, we see that each time we add two children to this scenario (the rest of the leaves being k-1) it is binary so there are three nodes involved, the leaf, the parent and the other leaf. Now the actual number of nodes is 2*(k-1) +3 = 2k+1 = 2(k+1)-1. We had set n = k+1 so this gives us 2n-1 for a new leaf scenario.

3. **Explain in detail that if m pointer fields are set aside in each node of a general m-ary tree to point to a maximum of m child nodes, and if the number of nodes in the tree is n, the number of null child pointer fields is n*(m-1)+1.**

An m-ary tree having m children max per node would have m*n child pointers, which let's say theoretically are null at first. Only the root node doesn't take a parent, so doesn't take a pointer, meaning that n-1 node is not being pointed to.

(m*n)-(n-1) = the amount of pointers that aren't being used. These are the null pointers.

= n*m-n+1

= n*(m-1)+1

4. **Define the Fibonacci binary tree of order n as follows: If n=0 or n=1, the tree consists of a single node. If n>1, the tree consists of a root, with the Fibonacci tree of order n-1 as the left subtree and the Fibonacci tree of order n-2 as the right subtree. Write a method that builds a Fibonacci binary tree of order n and returns a pointer to it.**

```
class Node:
         def __init__(self):
           self.left = None
           self.right = None


       def tree(n):
         if n == 0 or n == 1:
           n = Node()
           n.left = None
           n.right = None
         else:
           n = Node()
           n.left = tree(n-1)
           n.right = tree(n-2)
         return n
```

5. **Answer the following questions about Fibonacci binary tree defined in the previous problem.**
    a. **Is such a tree strictly binary?**

Yes this tree is strictly binary because every node has two children.

   b. **What is the number of leaves in the Fibonacci tree of order n?**

The number of leaves is 1 if n = 0 or 1, but otherwise in each order

Number of leaves in order n = (leaves in order n-1) + (leaves in order n-2)

   c. **What is the depth of the Fibonacci tree of order n?**

The depth is always n-1.