Problem Set 7

Justin Ely

615.202.81.FA15 Data Structures

20 October, 2015

1a)

- The oth level (root) node has exactly 0 ancestors
- the first level will have exactly 1 ancestor, the root node
- the second level will have exactly 2 ancestors (1 node at level 1 and the root)
- ...

Following this trend, a node at level n will have exactly n ancestors.

1b)

- A tree with 1 leaf has 1 node
- A tree with 2 leaves has 3 total nodes
- A tree with 3 leaves has 5 nodes
- A tree with 4 leaves has 7 nodes
- ...

Following this trend, a regular binary tree with n leaves has 2n-1 total nodes.

2)

3)

For a general m-ary tree, the total number of pointers set aside will be equal to n x m. However, each node n (apart from the root) will remove 1 null pointer from this count as the pointer will connect to it's parent.

$$
\begin{aligned}
N_{null} &= (n * m) - (n - 1) && (1)\\
&= (n * m) - n + 1 && (2)\\
&= n * (m - 1) + 1 && (3)
\end{aligned}
$$

4)

```
class Tree
    data = Array()

    def maketree(value)
        newNode = Node()

        newNode.value = value
        newNode.left = Null
        newNode.right = Null

        return newNode


    def traverse(Node)
        If not Node == null:
            traverse(Node.left)
            print Node.value
            traverse(Node.right)


    def setleft(value, parent)
        if parent.left != null:
            raise Error

        child = maketree(value)
        child.right = parent

        data[IndexOf(parent) * 2] = child


    def setright(value, parent)
        child = maketree(value)
        child.right = parent.right

        data[IndexOf(parent)*2 +1] = child
```

5a)

```
def fib(n)
    if n = 1:
        Here.left = 0
    else:
        Here.left = fib(n-1)
        Here.right = fib(n-2)
```

5b)

No, the 1 node will only have an n-1 child, and not an n-2 child.

5c)

An order N fibonacci tree will have fib(n-1) + fib(n-2) leaves.

5d)

A fibonacci tree of order n has depth n.