CSC148 fall 2013

recursive structures

week 5

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/148/F13/
416-978-5899

October 18, 2013
Outline
What can we figure out from what’s given?
Recursion exercise: Tower of Anne Hoy

```python
def toah(n: int, src: int, dest: int, inter: int) -> None:
    """
    Print how to move n>0 cheeses from src to dest using intermediate inter.
    """
    if n > 1:
        toah(n-1, src, inter, dest)
        Move n-1 src → inter
toah(1, src, dest, inter)
        Move 1 src → dest
        Move n-1 inter → dest
toah(n-1, inter, dest, src)
    else:
        Move initial 1 dest → print statement
        print ("Move disk on \$3 to tower \$3 ", format(src,
dest))
```

recursion, natural and otherwise

some repeating structures
broccoli structure contains copies of itself
terminology

- set of nodes (possibly with values or labels), with directed edges between some pairs of nodes

- One node is distinguished as root

- Each non-root node has exactly one parent.

- A path is a sequence of nodes $n_1, n_2, \ldots, n_k$, where there is an edge from $n_i$ to $n_{i+1}$.

- There is a unique path from the root to each node. In the case of the root itself this is just $n_1$, if the root is node $n_1$.

- There are no cycles — no paths that form loops.
more terminology

- **leaf**: node with no children

- **internal node**: node with one or more children

- **subtree**: tree formed by any tree node together with its descendants and the edges leading to them.

- **height**: Maximum path length in a tree, where the length of a path is the number of edges in it. **nb**: The length of a path is sometimes defined by the number of **nodes** in it, which makes it taller by 1.

- **arity, branching factor**: maximum number of children for any node.
pre-order traversal

Visit root, then pre-order left subtree, then pre-order right subtree
exercise: code for preorder traversal

"""
A TreeList is a Python list with 3 elements
  --- element 0 is a value
  --- element 1 is either a TreeList or None
  --- element 2 is either a TreeList or None
"""

def preorder(tl: 'TreeList') -> list:
    """
    Return list of values in tl in preorder
    """
    >>> T = [5, [4, None, None], [3, [2, None, None], [1, None, None]]]
    >>> preorder(T)
    [5, 4, 3, 2, 1]
    """
in-order traversal

Visit in-order left subtree, then root, then in-order right subtree
post-order traversal

Visit post-order left subtree, then post-order right subtree, then root