Announcements
Tree Class
Tree Review

Recursive description (wooden trees):  
A **tree** has a **root** value and a list of **branches**  
Each branch is a **tree**  
A tree with zero branches is called a **leaf**  

Relative description (family trees):  
Each location in a tree is called a **node**  
Each **node** has a **value**  
One node can be the **parent/child** of another
Tree Class

A Tree has a root value and a list of branches; each branch is a Tree

class Tree:
    def __init__(self, root, branches=[]):
        self.root = root
        for branch in branches:
            assert isinstance(branch, Tree)
        self.branches = list(branches)

def fib_tree(n):
    if n == 0 or n == 1:
        return Tree(n)
    else:
        left = fib_tree(n-2)
        right = fib_tree(n-1)
        fib_n = left.root + right.root
        return Tree(fib_n, [left, right])
def tree(root, branches=[]):
    for branch in branches:
        assert is_tree(branch)
    return [root] + list(branches)
def root(tree):
    return tree[0]
def branches(tree):
    return tree[1:]

(Demo)
Tree Mutation
Example: Pruning Trees

Removing subtrees from a tree is called *pruning*.

Prune branches before recursive processing.

```python
def prune(t, n):
    '''Prune sub-trees whose root value is n.'''
    t.branches = [b for b in t.branches if b.root != n]
    for b in t.branches:
        prune(b, n)

(Demo)
```
Example: Pruning Trees

Removing subtrees from a tree is called **pruning**

Prune branches before recursive processing

**Memoization:**
- Returned by `fib`
- Found in cache
- Skipped
Hailstone Trees
Hailstone Trees

Pick a positive integer $n$ as the start

If $n$ is even, divide it by 2

If $n$ is odd, multiply it by 3 and add 1

Continue this process until $n$ is 1

( Demo )

```python
def hailstone_tree(k, n=1):
    """Return a Tree in which the paths from the leaves to the root are all possible hailstone sequences of length $k$ ending in $n$."""

All possible $n$ that start a length-8 hailstone sequence

( Demo )
```