61A Lecture 19

Monday, March 9

---

**Announcements**

- Project 3 due Thursday 3/12 @ 11:59pm
- Project party on Tuesday 3/10 5pm-6:30pm in 2050 VLSB
- Bonus point for early submission by Wednesday 3/11
- Guerrilla section this weekend (details announced soon)
- Homework 6 due Monday 3/16 @ 11:59pm
- Midterm 2 is on Thursday 3/19 7pm-9pm
  - Fill out conflict form if you cannot attend due to a course conflict

---

**Measuring Efficiency**

**Recursive Computation of the Fibonacci Sequence**

Our first example of tree recursion:

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-2) + fib(n-1)
```

**Memoization**

Idea: Remember the results that have been computed before

```
def memo(f):
    cache = {}
    def memoized(n):
        if n not in cache:
            cache[n] = f(n)
        return cache[n]
    return memoized
```

**Memoized Tree Recursion**

Call to fib
- Found in cache
- Skipped

**Tree Class**

**Tree Class**

A tree has an entry (any value) at its root and a list of branches.

```python
class Tree:
    def __init__(self, entry, branches=()):
        self.entry = entry
        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)
        return Tree(left.entry + right.entry, [left, right])
```

**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

```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."""
```

**Binary Tree Class**

A binary tree is a tree that has a left branch and a right branch.

Idea: Fill the place of a missing left branch with an empty tree

Idea 2: An instance of BinaryTree always has exactly two branches

```python
class BinaryTree(Tree):
    empty.is_empty = True

def __init__(self, entry, left=empty, right=empty):
    Tree.__init__(self, entry, (left, right))

@property
def left(self):
    return self.branches[0]

@property
def right(self):
    return self.branches[1]
```