Tree Recursion
Announcements

• Midterm Grading: Out later this week
  – We are taking bugs into account that happened during the test.
  – We’ll have a 1 week window for regrades after scores are released.
• Clobber Policy:
  – You will be able to replace your midterm score with your final exam score if it’s higher. You don’t need to do anything to take advantage of this.
Learning Objectives

• Write Recursive functions with multiple recursive calls
• Understand Recursive Fibonacci
• Understand the quicksort algorithm
Tree Recursion:
Fibonacci
Tree Recursion

- Recursion which involves multiple recursive calls to solve a problem.
- Drawing out a function usually looks like an “inverted” tree.
Example I

List all items on your hard disk

- Files
- Folders contain
  - Files
  - Folders

```python
def process_directory(directory):
    for item in directory:
        if is_file(item):
            process_file(item)
        else:
            process_directory(item)
```
The Fibonacci Sequence

\[ \text{fibonacci}(n) = \text{fibonacci}(n-1) + \text{fibonacci}(n-2) \]

where \( \text{fibonacci}(1) = 1 \) and \( \text{fibonacci}(0) = 0 \)

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89…
Tree Recursion: Quicksort
Quicksort

- A fairly simple sorting algorithm
- Goal: Sort the list by breaking it into partially sorted parts
  - Pick a “pivot”, a starting item to split the list
  - Remove the pivot from your list
  - Split the list into 2 parts, a smaller part and a bigger part
  - Then recursively sort the smaller and bigger parts
  - Combine everything together: the smaller list, the pivot, then the bigger list
QuickSort Example

[3, 3, 1, 4, 5, 4, 3, 2, 1, 17]
[3, 1, 3, 2, 1]
[1, 3, 2, 1]
[1]
[1]  [3, 2]
[1]
[3, 1, 2, 1]
[1, 1, 2, 3]
[1, 1, 2, 3, 3]
[1, 1, 2, 3, 3, 4, 4, 5, 17]

[4, 5, 4, 17]
[4]
[5, 17]
[4]
[5, 17]
[4, 4, 5, 17]
Tree Recursion

• Break the problem into multiple smaller sub-problems, and Solve them recursively

```python
def split(x, s):
    return [i for i in s if i <= x], [i for i in s if i > x]

def quicksort(s):
    """Sort a sequence - split it by the first element,
    sort both parts and put them back together.""
    if not s:
        return []
    else:
        pivot = s[0]
        smaller, bigger = split(pivot, s[1:])
        return quicksort(smaller) + [pivot] + quicksort(bigger)

>>> quicksort([3,3,1,4,5,4,3,2,1,17])
[1, 1, 2, 3, 3, 3, 4, 4, 5, 17]
```