CSSE 230 Day 7

Recursion Again (and again …)

Check out from SVN: *Recursion* and *Trees* projects

Agenda

- Student questions about anything!

- Hardy/ColorizeFSM
- Better MCSS algorithm
- Recursion review
- Recursion programming exercise

Note: The next seven days are likely to be the busiest of the term in this course. Two medium-sized programs to write, and challenging written problems. Start early (especially on the programming projects).

Hardy Part 2

- Do a slightly different Hardy calculation
- With certain space constraints
- Make it as fast as you can without violating the problem constraints
  - Mainly, that you can make no pre-assumptions about the sizes of the numbers other than that they are smaller than Java's longest long integer
- Carefully select data structures to use
- When you can correctly find n^{th} Hardy numbers, you are probably halfway done
  - Then comes efficiency

ColorizeFSM

- Lots of tools for writing to the html.
- One person already finished it.
- How should we implement the FSM?
  - 3 choices

Possible Representations of the Finite State Machine

Q7

Diagrams on the whiteboard

- 2-Dimensional array:
  - Rows indexed by state, Columns by input character.
  - Each array entry is a pair object (as in DS Section 3.7):
    - [next state, what to print]
- Monolithic controller with nested switch statements
- Have a class for each state, that implements the State interface.
  - Choice # 1may have shorter code
  - Choice #2 is probably easier to write and modify
  - Choice #3 is most modular and aesthetic! We like it!

You’ve met your partner

- Plan when you'll be working
- Pair programming, but I suggest that each of you take the "research lead" for one of the programs
- Begin thinking about both

Q1-3

Weiss's Recursion Principles

1. **Base Case:** Always have at least one case that can be solved without recursion.
2. **Make Progress:** Every recursive call must progress toward some base case.
3. **"You gotta believe"**: Always assume that the recursive call does what it is supposed to do.
   - Use that result in building the "higher-level" solution

Q4

Recursive List Size

```
public class ListNode<T> {
   T element;
   ListNode<T> next;

   public ListNode(T e,
         ListNode<T> n) {
      this.element = e;
      this.next = n;
   }

   public ListNode(T e) {
      this(e, null);
   }

   public ListNode() {
      this(null, null);
   }
}
```

```
public class LinkedList<T> {
private ListNode<T> head,
private ListNode<T> tail;


// lots of other stuff.
// Write a size() method.






}
```

Fibonacci Numbers

- Each Fibonacci number (except the first two) is the sum of the previous two Fibonacci numbers.

i	0	1	2	3	4	5	6	7	8
F_i	0	1	1	2	3	5	8	13	21

- $F_0=0,\ \ F_1=1,\ \ F_{i+2} = F_i + F_{i+1}$

```
public static int fib(int n) {
        if (n < 2)
                return n;
        return fib(n-2) + fib(n-1);
}
```

The Trouble with Fib

- Easy to program!
  Expensive!

```
public static int fib(int n) {
      if (n < 2)
            return n;
      return fib(n-2) + fib(n-1);
}
```

Q5, Q6

Weiss's Fourth Recursion Principle

- **Compound Interest rule**: Don't recursively recompute the same things over and over in separate recursive calls.
- Alternatives:
  - Cache previously computed values in an array (memoization)
  - Use a loop
- This is a reminder from 220/221.

Q7

Recursive ParseInt?

- Input: a string representation of a positive integer
- Output: the integer
- …using recursion

Recursive binary search is elegant

- Input: an array of integers and an element for which to search.
- Output: the index where it was found.
  - -1 if not found
- Big-Oh runtime of binary search?

Famous Diversion – Towers of Hanoi (a relevant interlude)

- The Towers of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883.
- We are given a tower of disks initially stacked in decreasing size on one of three pegs
- The objective is to transfer the entire tower to one of the other pegs,
- moving only one disk at a time and
- never placing a larger disk on top of a smaller disk

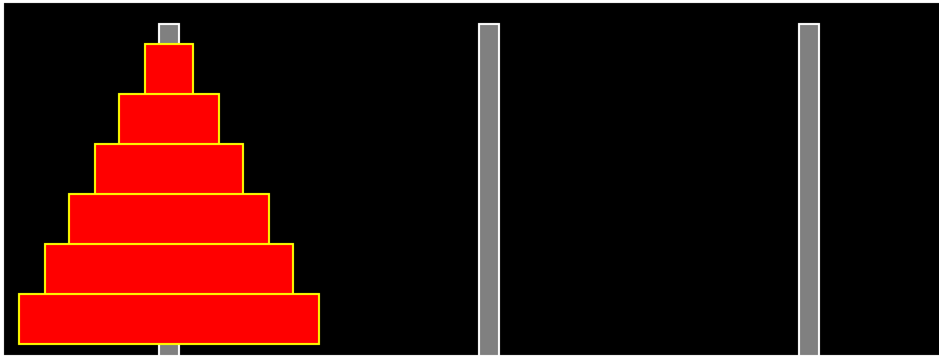


Image is from

http://www.cut-the-knot.com/recurrence/hanoi.html

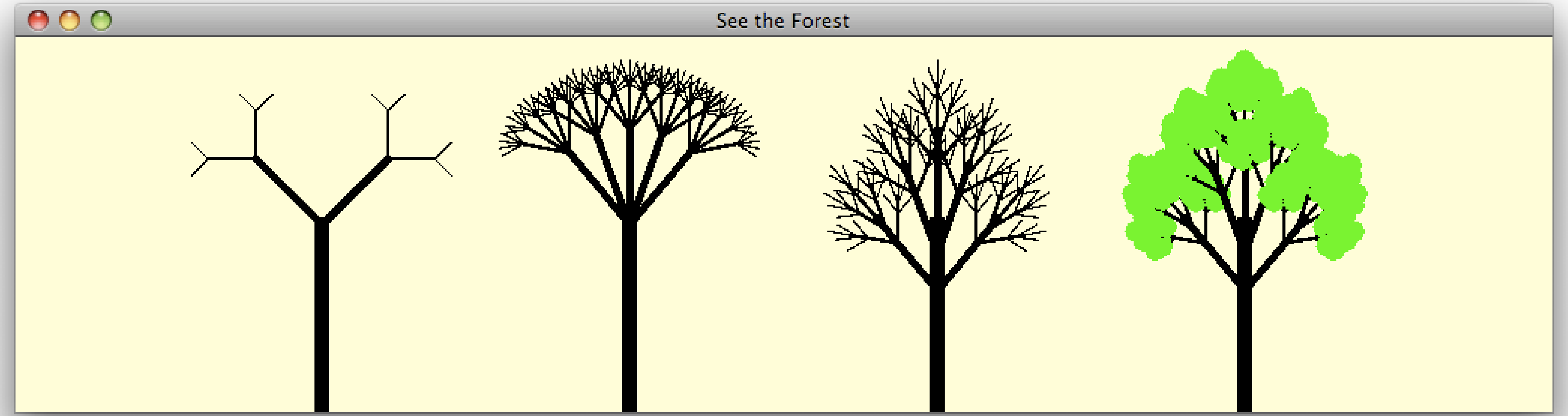
Towers of Hanoi – hands on

Demo!

Towers of Hanoi

- Write the method (and its recursive helper)
- Analyze it: count the total moves required to move n disks from one peg to another
  - I.e., write and solve the recurrence relation

Trees



- Read assignment linked from schedule, WA3
- Check out *Trees* project from individual SVN repository
- We will look at the code together