



CSSE 220

Day 20

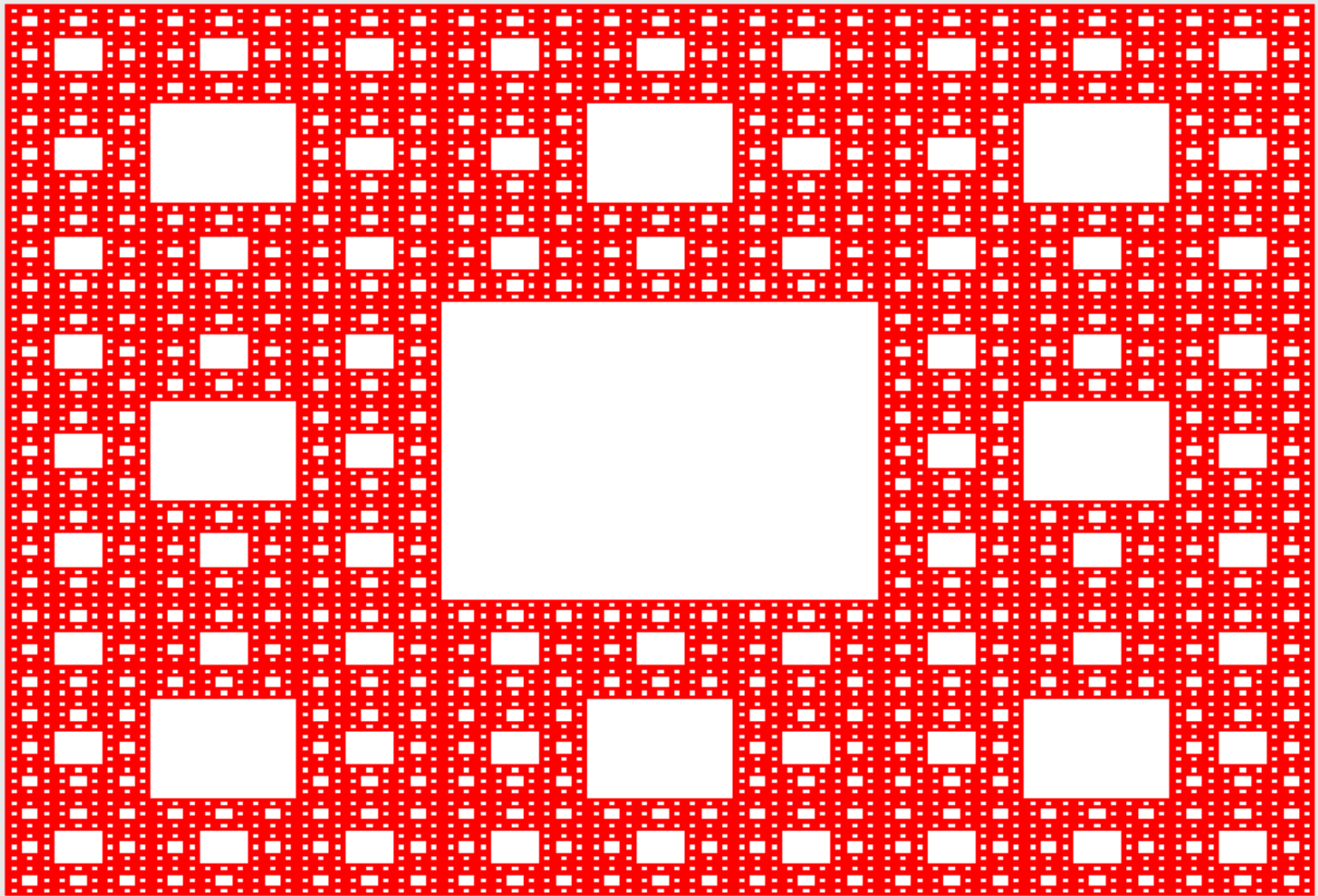
Recursion, Efficiency, and
the Time-Space Trade Off;
Selection Sort and Big-Oh

Checkout *Recursion2* project from SVN

Key Rules to Using Recursion

- ▶ Always have a **base case** that **doesn't recurse**
- ▶ Make sure recursive case always makes **progress**, by **solving a smaller problem**
- ▶ **You gotta believe**
 - Trust in the recursive solution
 - Just consider one step at a time

Sierpinski Carpet



Frames for Tracing Recursive Code

1. Draw box when method starts

2. Fill in name and first line no.

3. Write class name (for static method) or draw reference to object (for non-static method)

method name, line number

scope box

parameters
and local variables

4. List every parameter and its argument value.

5. List every local variable declared in the method, **but no values yet**

6. Step through the method, update the line number and variable values, draw new frame for new calls

7. "Erase" the frame when the method is done.

Thanks for
David Gries for
this technique

Q1-2

What the Fib?

- ▶ Why does recursive Fibonacci take so long?!?
- ▶ Can we fix it?

Classic Time–Space Trade Off

- ▶ A deep discovery of computer science
- ▶ In a wide variety of problems we can tune the solution by varying the amount of storage space used and the amount of computation performed
- ▶ Studied by “Complexity Theorists”
- ▶ Used everyday by software engineers

Mutual Recursion

- ▶ Two or more methods that call each other repeated

Example

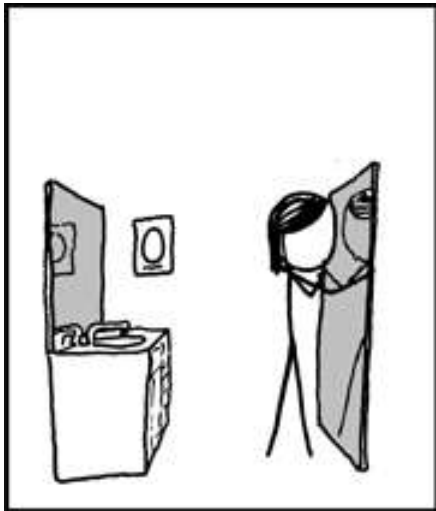
- ▶ Hofstadter Female and Male Sequences:

$$F(n) = \begin{cases} 1 & \text{if } n = 0 \\ n - M(F(n - 1)) & \text{if } n > 0 \end{cases}$$

$$M(n) = \begin{cases} 0 & \text{if } n = 0 \\ n - F(M(n - 1)) & \text{if } n > 0 \end{cases}$$

- ▶ Questions:
 - How often are the sequences different in the first 50 positions? first 500? first 5,000? first 5,000,000?

Two Mirrors



If you actually do this, what really happens is Douglas Hofstadter appears and talks to you for eight hours about strange loops.

Team Preferences Survey

- ▶ Starting team project Friday
- ▶ Need some input:
 - Log on to ANGEL
 - Go to course ANGEL page
 - Navigate to **Lessons → Project Forms → Vector Graphics Team Preferences**
 - Complete the short survey


What is sorting?

»» Let's see...

Why study sorting?

»» Shlemiel the Painter

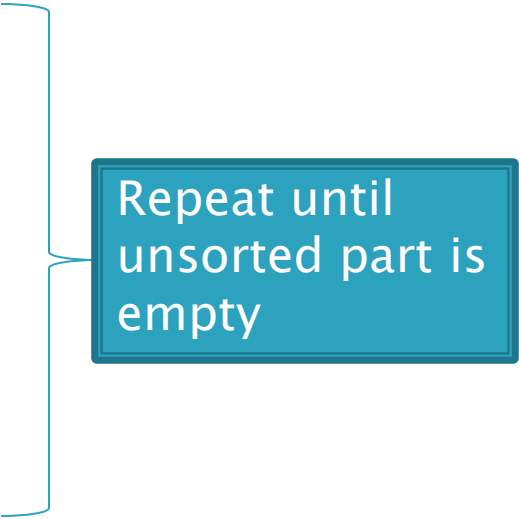
Course Goals for Sorting: You should...

- ▶ Be able to **describe** basic sorting algorithms:
 - Selection sort
 - Insertion sort
 - Merge sort
 - Quicksort
 - ▶ Know the **run-time efficiency** of each
 - ▶ Know the **best and worst case** inputs for each
- 

Selection Sort

- ▶ Basic idea:

- Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)
- Find the smallest number in the unsorted part
- Move it to the end of the sorted part (making the sorted part bigger and the unsorted part smaller)



Repeat until
unsorted part is
empty

Profiling Selection Sort

- ▶ **Profiling**: collecting data on the run-time behavior of an algorithm
- ▶ How long does selection sort take on:
 - 10,000 elements?
 - 20,000 elements?
 - ...
 - 80,000 elements?


Analyzing Selection Sort

- ▶ **Analyzing**: calculating the performance of an algorithm by studying how it works, typically mathematically
- ▶ Typically we want the **relative** performance as a function of input size
- ▶ Example: For an array of length n , how many times does **selectionSort()** call **compareTo()**?

Handy Fact

$$1 + 2 + \dots + (n - 1) + n = \frac{n(n + 1)}{2}$$

Big-Oh Notation

- ▶ In analysis of algorithms we care about differences between algorithms on very large inputs
 - ▶ We say, “selection sort takes on the order of n^2 steps”
 - ▶ Big-Oh gives a formal definition for “on the order of”
- 

Formally

- ▶ We write $f(n) = O(g(n))$, and say “ f is big-Oh of g ”
- ▶ if there exists positive constants c and n_0 such that
- ▶ $0 \leq f(n) \leq c g(n)$
for all $n > n_0$
- ▶ g is a **ceiling** on f

