

CSSE 220

Day 22

Sierpiński, Recursion and
Efficiency, Mutual Recursion

Checkout *Recursion2* and
SierpinskiTriangle projects from SVN

Notes

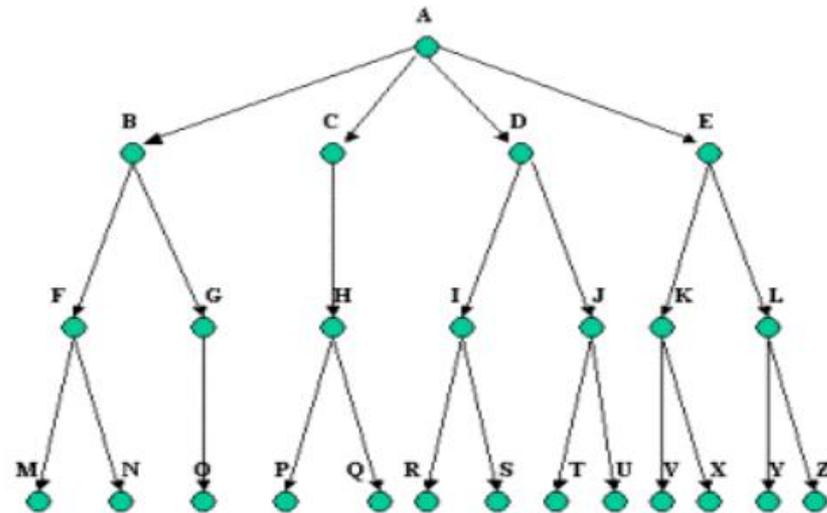
- ▶ Quiz Grading
 - 1.5 Quizzes left
 - Should get back by Wednesday
- ▶ Exam Review – Wednesday
 - 1st hour for exam review
 - 2nd hour for work time (5th hour will be TA only)
- ▶ Exam
 - Written Portion for first half on Friday
 - Programming Portion on Monday

Recap: What are recursive methods?

- ▶ Any method that calls itself
 - On a simpler problem
 - So that it makes progress toward completion
 - **Indirect recursion:** May call another method which calls back to it.

When should recursive methods be used?

- ▶ When implementing a recursive definition
- ▶ When implementing methods on recursive data structures



- ▶ Where parts of the whole look like smaller versions of the whole

The pros and cons of recursive methods

▶ The pros

- easy to implement,
- easy to understand code,
- easy to prove code correct

▶ The cons

- Sometimes takes more space and time than equivalent iterative solution
- Why?
 - because of function calls

Recap: 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

Can one little Fib hurt?

- ▶ Why does recursive Fibonacci take so long?!?

```
private static long fib(int n) {  
    // TODO: Convert this to use memoization.  
    long f;  
    if (n <= 2) {  
        f = 1;  
    } else {  
        long fNMOne = fib(n - 1);  
        long fNMTwo = fib(n - 2);  
        f = fNMOne + fNMTwo;  
    }  
    return f;  
}
```

- ▶ Can we fix it?

Memoization

- ▶ Save every solution we find to sub-problems
- ▶ Before recursively computing a solution:
 - Look it up
 - If found, use it
 - Otherwise do the recursive computation

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

- ▶ 2 or more methods call each other repeatedly
 - E.g., Hofstadter Female and Male Sequences

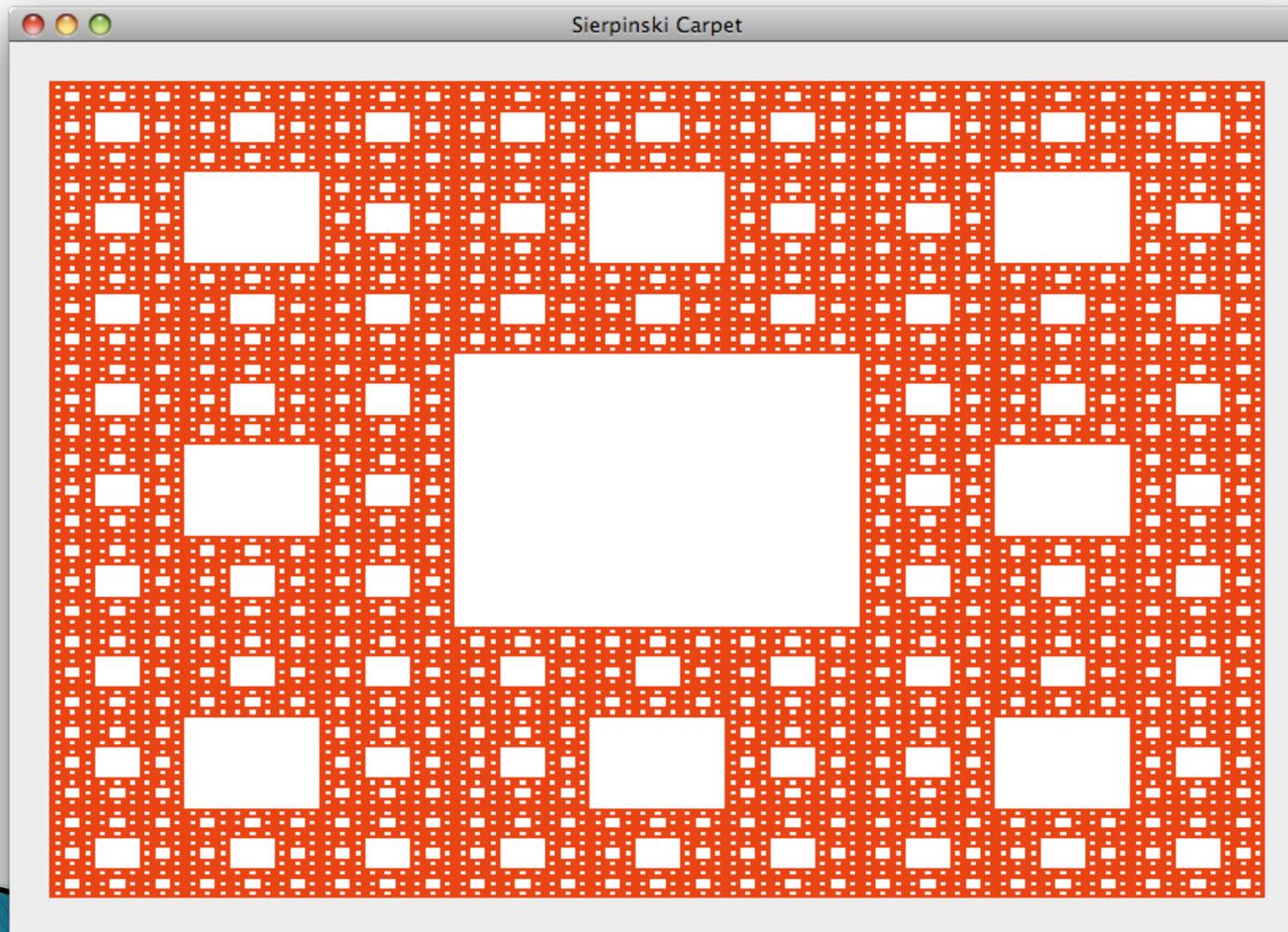
$$F(n) = \begin{cases} 1 & \text{if } n = 0 \\ n - M(F(n - 1)) & \text{if } n > 0 \end{cases}$$

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

- In how many positions do the sequences differ among the first 50 positions? first 500? first 5,000? first 5,000,000?

http://en.wikipedia.org/wiki/Hofstadter_sequence

In Class Work Time



Questions?

