Recursion continued

Dictionary lookup

- Suppose you’re looking up a word in the dictionary (paper one, not online!)
- You probably won’t scan linearly thru the pages – inefficient.
- What would be your strategy?
Binary search

binarySearch(dictionary, word) {
    if (dictionary has one page) { // base case
        scan the page for word
    }
    else { // recursive case
        open the dictionary to a point near the middle
determine which half of the dictionary contains word
        if (word is in first half of the dictionary) {
            binarySearch(first half of dictionary, word)
        } else {
            binarySearch(second half of dictionary, word)
        }
    }
}

Write a method binarySearch that accepts a sorted array of integers and a target integer and returns the index of an occurrence of that value in the array.
- If the target value is not found, return -1

| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|-------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| value | -4| 2 | 7 | 10| 15| 20| 22| 25| 30| 36| 42 | 50 | 56 | 68 | 85 | 92 | 103|

int index = binarySearch(data, 42); // 10
int index2 = binarySearch(data, 66); // -1
Recursive Algorithms

Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.

Try to find the pattern by cases

- One disk is easy
- Two disks...
- Three disks...
- Four disk...
Recursive Algorithms

Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.
Recursive Algorithms

Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.

Let's go play with it at:  http://www.mazeworks.com/hanoi/index.htm
Fibonacci’s Rabbits

- Suppose a newly-born pair of rabbits, one male, one female, are put on an island.
  - A pair of rabbits doesn’t breed until 2 months old.
  - Thereafter each pair produces another pair each month
  - Rabbits never die.
- How many pairs will there be after \( n \) months?

Do some cases, see a pattern?

<table>
<thead>
<tr>
<th>( n )</th>
<th>Young</th>
<th>Mature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

image from: http://www.jimloy.com/algebra/fibo.htm
The pattern...

m0: 1 young                   1
m1: 1 mature                   1
m2: 1 mature 1 young           2
m3: 2 mature 1 young           3
m4: 3 mature 2 young           5

\[ m_n = m_{n-1} \text{ (rabbits never die) } + \]
\[ m_{n-2} \text{ (newborn pairs) } \]

How fast does this rabbit population grow?

Fibonacci numbers

- The *Fibonacci numbers* are a sequence of numbers \( F_0, F_1, \ldots, F_n \) defined by:

\[
F_0 = F_1 = 1
\]

\[
F_i = F_{i-1} + F_{i-2} \text{ for any } i > 1
\]

- Write a method that, when given an integer \( i \), computes the \( nth \) Fibonacci number.
Fibonacci numbers

- recursive Fibonacci was expensive because it made many, many recursive calls
  - fibonacci(n) recomputed fibonacci(n-1, ... ,1) many times in finding its answer!
  - this is a common case of "overlapping sub-problems", where the sub-tasks handled by the recursion are redundant with each other and get recomputed

Fibonacci code

- Let's run it for \( n = 1, 2, 3, \ldots 10, \ldots, 20, \ldots \)
- What happens if \( n = 5, 6, 7, 8, \ldots \)
- Every time \( n \) increments with 2, the call tree more than doubles..
Growth of rabbit population

1 1 2 3 5 8 13 21 34 ...

every 2 months the population at least **DOUBLES**

Fractals – the Koch curve
Simpler example: cCurve

Can you draw the next one???

Other growth phenomenon: cCurve

The next one is: the previous one rotated left plus the previous one rotated right
/** Recursive function which draws a CCurve
 * @param rank of the CCurve
 * @param angle initial angle of the CCurve
 */

public static void cCurve(int rank, int angle) {
    if (rank <= 0) {
        addLine(angle);
    } else {
        cCurve(rank - 1, angle - 45);
        cCurve(rank - 1, angle + 45);
    }
}