



Recursion

How TO teach
and NOT TO teach
Recursion
(opinions from the experts?)

Fran Trees
Drew University
ftrees@drew.edu

Where do we find materials and examples to present in our lectures?

- Find some really smart people and ask them for help!

[Where do we find those really smart people?]

Motivated by AP EDG comments:

- ...And we teach about real world examples like bank-accounts and employees and Towers-Of-Hanoi.....
- Just say
 - **NO** to bank accounts
 - **NO** to employees ...
 - **NO** to Towers-Of-Hanoi?????

[Bank Accounts?]

- **NO** to bank accounts?
- What about using bank accounts to talk about recursion? After all...
 - a bank account (the prototypical object) has overdraft protection provided by another account (which may itself have overdraft protection, etc.).

[**NO** to Towers-Of-Hanoi?]

- about **162,000** hits for **Towers of Hanoi** on Google!

[NO to Towers-Of-Hanoi?]

- "...And, yes, please, no bank accounts, no employees, no barking dogs. (But let's keep Towers of Hanoi -- for their sentimental value.)"
- "...I will also echo the pitch to keep Towers of Hanoi, but not for sentimentality. I say keep it because it is a concrete, physical model for recursion that makes sense to kids. They can see and feel the patterns as they move disks, and this makes it pedagogically appropriate."

[No to Towers-Of-Hanoi?]

- Let me cast a vote for letting Towers of Hanoi finally die. I have never seen this as a helpful example when trying to explain recursion.

[Easy to understand?]

```
static void towers(char fromPeg, char toPeg, char tempPeg, int
    numDisks)
{
    if (numDisks == 1)
    {
        System.out.println(numDisks + "\t" + fromPeg + "\t"
            + toPeg);
    }
    else
    {
        towers(fromPeg, tempPeg, toPeg, numDisks-1);
        towers(fromPeg, toPeg, tempPeg, 1);
        towers(tempPeg, toPeg, fromPeg, numDisks-1);
    }
}
```

[Towers of Hanoi]

■ History:

- <http://www.lawrencehallofscience.org/Java/Tower/towerhistory.html>
- <http://www.dcs.napier.ac.uk/a.cumming/hanoi/rechelp.html>
- <http://www.lhs.berkeley.edu/Java/Tower/index.html>

■ Explanation of Algorithm

- <http://www.dcs.warwick.ac.uk/~sgm/open/hanoi.html>

Towers of Hanoi

■ Solution:

- <http://www.cise.ufl.edu/~sahni/dsaaj/JavaVersions/applications/TowersOfHanoi/TowersOfHanoi.htm>
- <http://www.ifors.ms.unimelb.edu.au/tutorial/hanoi/>

■ Problem and Solution

- <http://www.mazeworks.com/hanoi/>
- <http://javascript.internet.com/games/hanoi.html>
- <http://www.lhs.berkeley.edu/Java/Tower/Tower.html>

[NO to Towers-Of-Hanoi?]

- Personally, I HATE Towers-Of-Hanoi!

but....

[From a workshop participant!]

- "At any rate, I'm at a loss--I completely forget the shortcut method for solving the Towers of Hanoi in the least number of moves possible.
- I know it was "_____ " then "make a legal move", over and over. I CAN'T REMEMBER THE FIRST PART!!!!"



The three rules of Recursion:

Approach every recursive problem as if it were a journey; if you follow these rules, you will complete the journey successfully.

- RULE 1: Find out how to take just one step.
- RULE 2: Break each journey down into one step plus a smaller journey.
- RULE 3: Know when to stop.



After all...Recursive Solutions

are easier to understand than
iterative solutions.

[Recursive solutions are easier to understand than Iterative solutions.]

- Iterative

```
for(i=0; i<=10; i++)  
    System.out.println(i);
```

[Recursive solutions are easier to understand than Iterative solutions..]

- Recursive

```
public static void printNumber(int n)
{
    if (n >= 0)
    {
        printNumber(n-1);
        System.out.println(n);
    }
}
```

Recursive solutions are easier to understand than Iterative solutions.

- Iterative

```
public static boolean palindrome(String s)
{
    String reverse = "";
    for(int i = s.length()-1; i>=0; i--)
    {
        reverse += s.charAt(i);
    }
    return(reverse.equals(s));
}
```

[Recursive solutions are easier to understand than Iterative solutions.]

■ Recursive

```
public static boolean palindrome(String s)
{
    if (s.length() <= 1)
        return true;
    if (s.charAt(0) == s.charAt(s.length()-1))
        return palindrome(s.substring(1, s.length()-1));
    else return false;
}
```

[Non-Motivating Examples...]

- Fibonacci numbers
- Factorial
- Power
- Combinations $C(n,k)$

but...

[Combinations]

```
public static int combo(int amount,int value)
{
    if ((amount < 0) || (value == 0))
        return(0);

    else if (amount == 0)
        return( 1);

    else
        return( combo(amount,value - 1) + combo(amount - value, value));
}
```

Evaluate `combo(4,3)`

(4, 3)
need help...

[Evaluate: acker (2 , 3)]

$$\text{acker}(x, y) = \begin{cases} y + 1 & \text{when } x = 0 \\ \text{acker}(x - 1, 1) & \text{when } x \neq 0, y = 0 \\ \text{acker}(x - 1, \text{acker}(x, y - 1)) & \text{when } x \neq 0, y \neq 0 \end{cases}$$

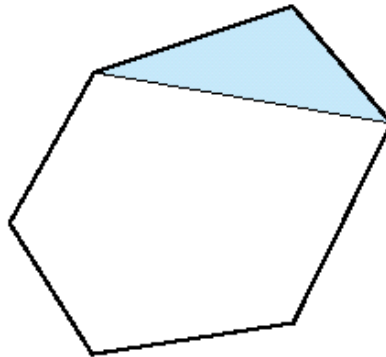
Perhaps more useful but
maybe not motivating...

- Print the binary for of a decimal number.
- Return the Greatest Common Factor of two numbers (Euclid's Algorithm).
 - $\text{gcd}(28, 8) \quad 28 = 3 * 8 + 4$
 - $\text{gcd}(28, 8) = \text{gcd}(8, 4)$
 - $\text{gcd}(8, 4) \quad 8 = 2 * 4 + 0$
 - $\text{gcd}(8, 4) = \text{gcd}(4, 0)$
 - $\text{gcd}(4, 0) = 4$

Area of a Polygon

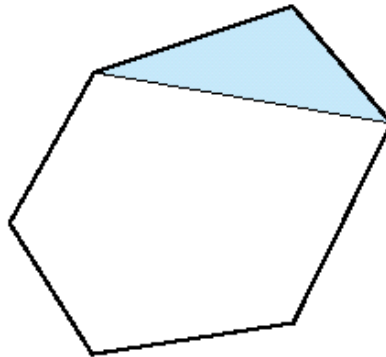
To compute the area of a polygon with more than three corner points:

- chop off a triangle and determine the area of the triangle (take one step).
- compute the area of the remaining polygon (a smaller journey)



[Maybe a bit more motivating...]

$$A = \frac{|x_1 * y_2 + x_2 * y_3 + x_3 * y_1 - y_1 * x_2 - y_2 * x_3 - y_3 * x_1|}{2}$$



[Tracing Recursive Problems]

- Affectionately known as "mystery methods" in sample AP problems.

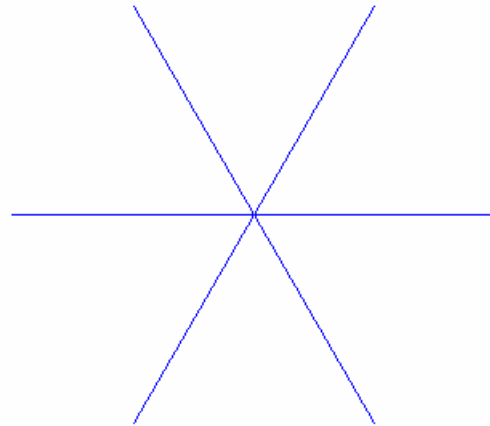
What makes problems motivating?

- Graphics

[Motivating example?]

- What about a star (really 6 lines)?

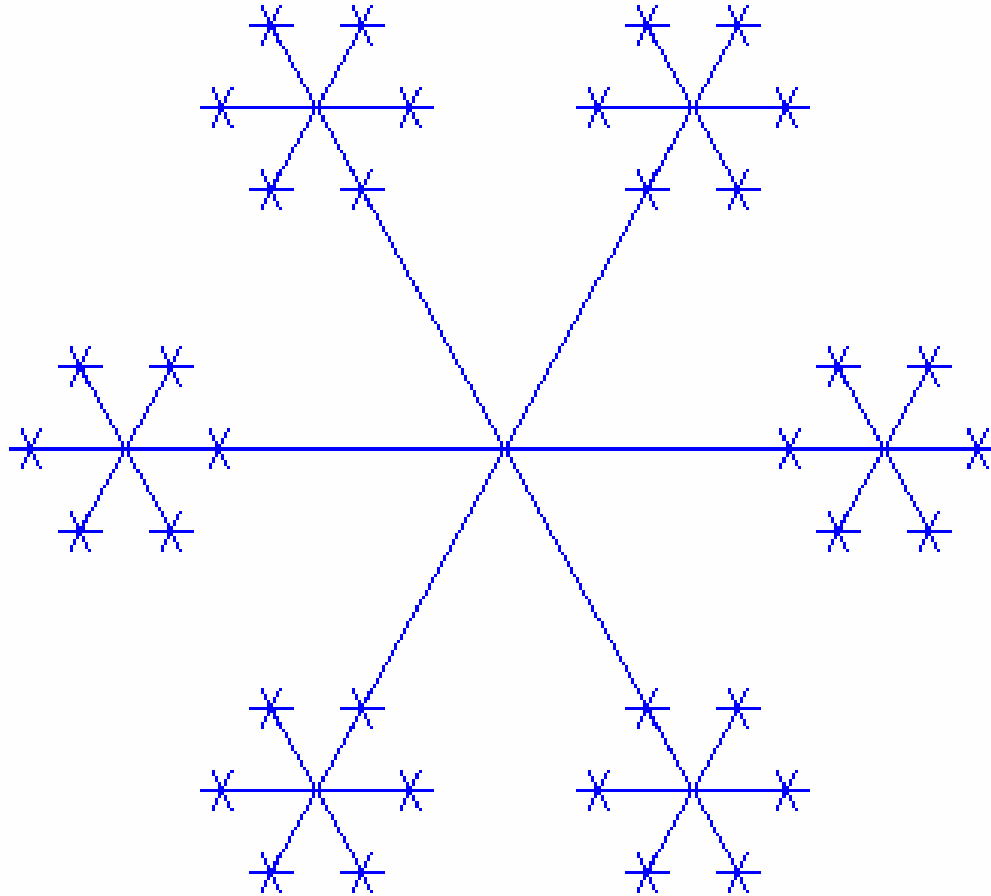
```
drawLine( cx, cy, endX, endY )
```



- Perhaps not so much...

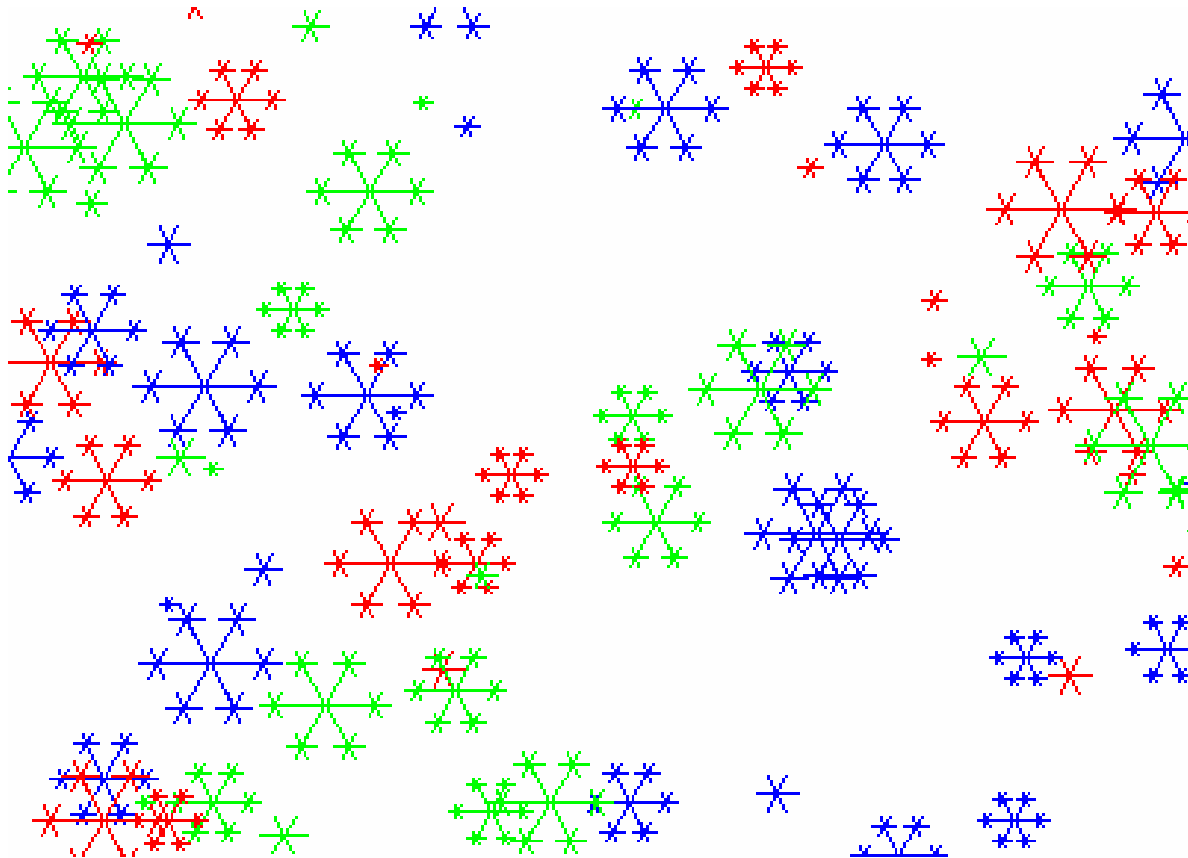
http://chortle.ccsu.ctstateu.edu/CS151/Notes/chap74/ch74_7.html

[What about a snowflake?]

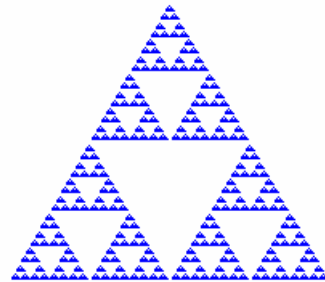


http://chortle.ccsu.ctstateu.edu/CS151/Notes/chap74/ch74_9.html

Or many snowflakes?



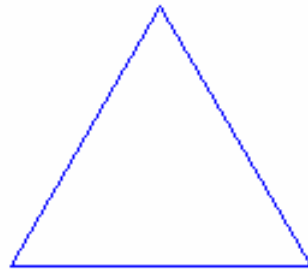
Sierpinski Triangle



- <http://www.arcytech.org/java/fractals/sierpinski.shtml>
- <http://math.bu.edu/DYSYS/chaos-game/node2.html>
- <http://math.rice.edu/~lanius/fractals/sierjava.html>

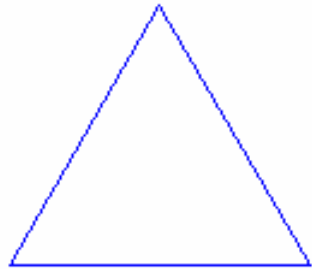
Sierpinski Triangle

- Draw a triangle.
 - How do we do this?



[Sierpinski Triangle]

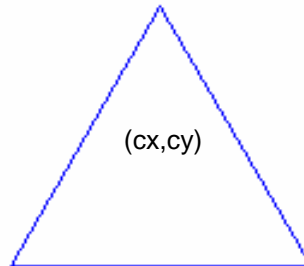
- Now what?



<http://www.arcytech.org/java/fractals/sierpinski.shtml>

Sierpinski Triangle

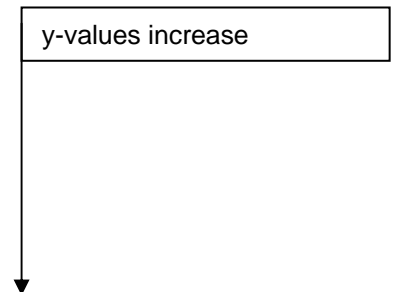
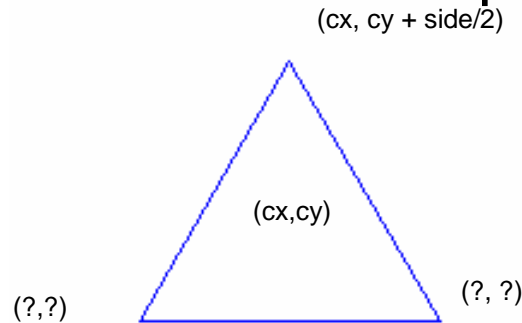
- If the center of this triangle is at (cx, cy) , and the length of a side is $side$, find the coordinates of the 3 vertices.



<http://www.arcytech.org/java/fractals/sierpinski.shtml>

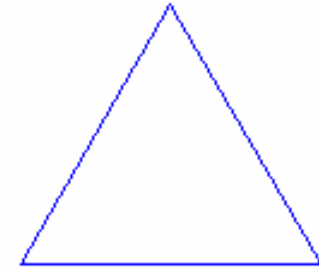
Sierpinski Triangle

- If the center of this triangle is at (cx, cy) , and the length of a side is $side$, find the coordinates of the 3 vertices.
 - (Use 30-60-90 relationships)



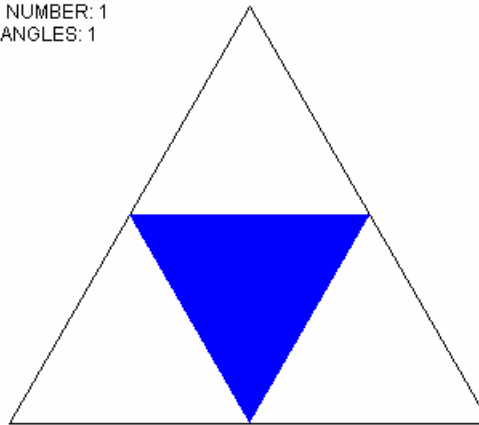
<http://www.arcytech.org/java/fractals/sierpinski.shtml>

[Sierpinski Triangle



- Now what?
 - If the side is small, draw one triangle.
 - If the size is large (not small), draw three triangles that are half the size.

ITERATION NUMBER: 1
TOTAL TRIANGLES: 1

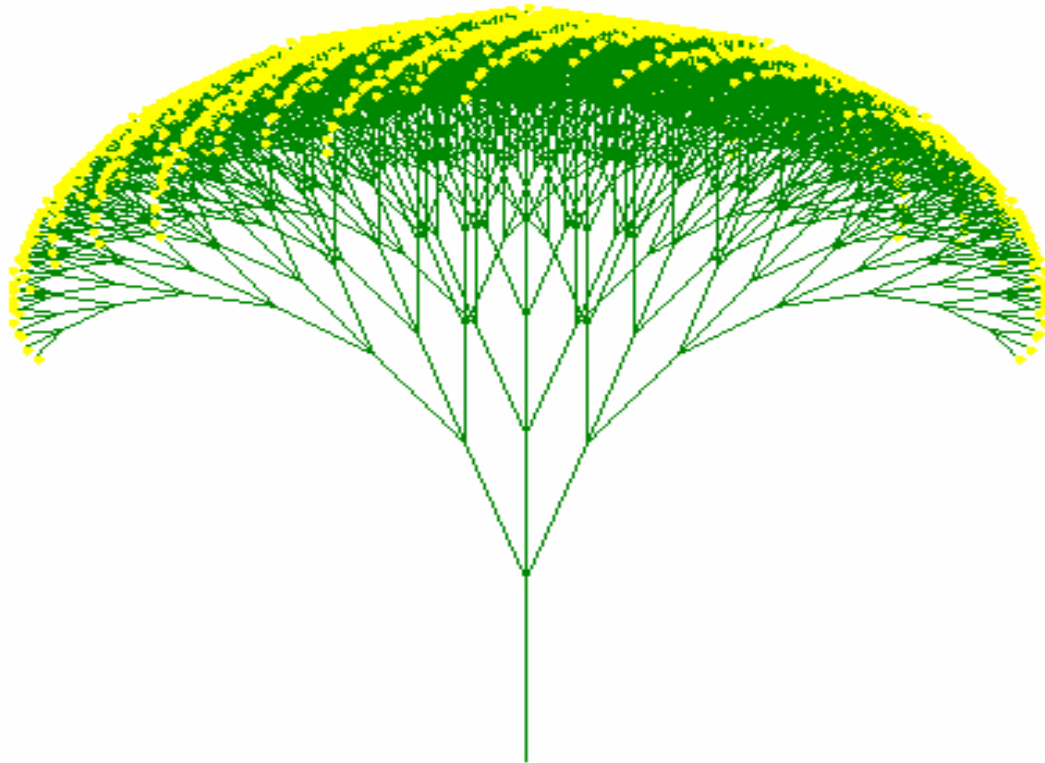


[Need more triangle?]

- Koch's Snowflake

<http://www-cs-faculty.stanford.edu/~eroberts/jtf/demos/KochSnowflakeDemo.html>

[And of course there's broccoli.]



Introducing Recursion

- The Cat in the Hat Comes Back
 - Dr. Seuss
- Martin and the Dragon
 - Touretzky, David S. Touretzky. *Common Lisp: A Gentle Introduction to Symbolic Computation*. (Chapter 8)

Dealing with Infinite Recursion

- Martin and the Dragon
 - Touretzky, David S. Touretzky. *Common Lisp: A Gentle Introduction to Symbolic Computation*. (Chapter 8)

Dealing with Infinite Recursion

- There's a hole in the bucket....
 - http://www.enchbyench.com/angie/dear_iza.htm
 - http://www.songsforteaching.com/folk/the_resaholeinthebucket.htm

[The ideal educational situation:]

- Teach what you need when you need it.

A Story....

- Late one night, a man approaches a street corner and sees another man crawling around on his hands and knees under a lamppost.

First Man: "What are you doing?"

Second Man: "Looking for my lost watch. Can you help?"

So the two of them crawl around and look for awhile, finding nothing.

First Man: "It doesn't seem to be here. Are you sure this is where you lost it?"

Second Man: "No. I lost it in that alley over there."

First Man: "Then why are you looking for it under the lamppost?"

Second Man: "The light is better here."

■ Posted by Jim Huggins, Wednesday, November 24, 2004 8:47 AM

Thanks to my friends...

- Kim Bruce
- Eric Roberts
- Stuart Reges
- Mike Clancy
- Susan Horwitz
- Owen Astrachan
- Gary Litvin
- Maria Litvin
- Roselyn Teukolsky

[Thanks to the EDG!]

- Jim Huggins
- Carl G. Alphonse
- Greg King

References

- Kjell, Bradley. *Introduction to Computer Science*
<http://chortle.ccsu.ctstateu.edu/CS151/cs151java.html#17>
- Roberts, Eric. *Thinking Recursively*.
<http://www.acm.org/education/jtf/>
- Bruce, Kim, Andrea Danyluk, and Thomas Murtagh. *Why Structural Recursion should be taught before Arrays in CS1*.
<http://www.iol.ie/~jmchugh/csc302/>
- <http://www.cs.bham.ac.uk/resources/courses/java/msc/lectures/lecture1-2.pdf>
- Touretzky, David S. Touretzky. *Common Lisp: A Gentle Introduction to Symbolic Computation*. (Chapter 8) <http://www-2.cs.cmu.edu/~dst/LispBook/>
- Trees, Frances P. and Cay Horstmann. *Computing Concepts with Java Essentials AP CS Student Guide*.

[Finally....]

- In order to understand recursion you must understand recursion.

Dan R. Ghica, *Java Workshop: Recursion*

<http://www.cs.bham.ac.uk/resources/courses/java/msc/lectures/lecture1-2.pdf>