Recursion

- what is it?
- how to build recursive algorithms
- recursion analysis
- tracing simple recursive functions
- hands on attempts at writing simple recursion
- examples

Recursion

- it's a problem solving technique where an algorithm is defined in terms of itself
- a recursive method is a method that calls itself
- breaks down the input and applies the same logic to a smaller and smaller piece of the problem until it is solvable without recursion.

Recursion vs. Iteration

- in general, any algorithm that is implemented using a loop can be transformed into a recursive algorithm

- moving in the reverse direction is not always possible!

Recursion Analysis

- in general, recursive algorithms are
  - more efficient (size of source code)
  - more readable (but occasionally quite the opposite!)
  - more “elegant”
  - Simpler because OS runtime stack does the bookkeeping of the various states of the data
- side effects
  - Expensive replication of memory
  - Other “over head” costs of the OS

Recursion Components

- Solution to the "base case" problem
  - for what values can we solve without another recursive call?'
- Reducing the search space
  - modify the value so it is closer to the base case
- The recursive call
  - Where do we make the recursive call?
  - What do we pass into that call?

An Example -- GCD

- Greatest Common Divisor -- GCD
- largest integer that can divide two other integers
- very old algorithm, devised between 400 and 300 B.C. by Euclid

GCD Algorithm

given two positive integers X and Y,

where X >= Y, the GCD(X,Y) is

- equal to Y if X mod Y is zero
- equal to the GCD(Y, X mod Y) if X mod Y > 0
- Notice the algorithm only terminates when the X % Y is zero.
- Notice that each time the function calls it self, the 2nd argument gets closer to zero and must eventually reach zero.

What is the output of this code?

```
public void foo(int x){
   if (x == 0)
          return;
   else
          System.out.println(x);
          foo(x - 1);
}

foo(7);
```

identify the base case, recursive call and reduction of the input toward the base case.

What is the output of this code?

```
public int foo(int x){
   if (x == 0)
            return 0;
   else
            return (x + foo(x-1));
}

Sytem.out.print(foo(7));
```

identify the Base case, recursive call and modification of the input toward the base case

What is the output of this code?

```
public static mystery(int x, int y)
{
    if(x ==0 || y == 0)
        return 0;
    else
        return x + mystery(y -1 , x);
}
```

Now.. You help me write this

- Write a recursive function that accepts an int and prints that integer out in reverse on one line

- What is the base case ?
- How do I reduce the input toward base case ?
- What do I pass to the recursive call ?

One more try!

- Write a recursive function that accepts a string and prints that string out in reverse on one line.

- What is the base case ?
- How do I reduce the input toward base case ?
- What do I pass to the recursive call ?

Other Examples ...

- Bad examples
  - factorial
  - exponential
  - Fibonacci numbers
  - power

Other Examples ...

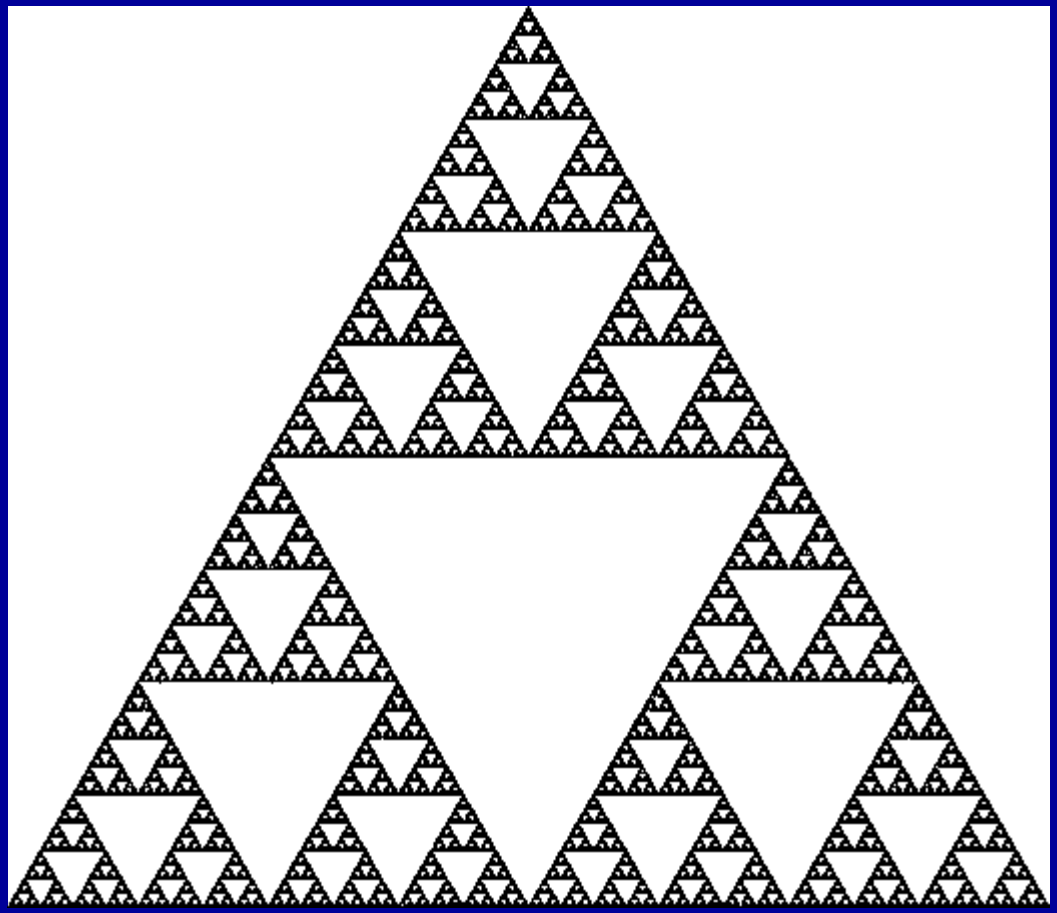
- Good examples
  - Towers of Hanoi
  - Traversing a binary tree
  - Inserting in order into a list
  - Maze traversal (backtracking)
  - Eight Queens (backtracking)

Sierpinski triangle

- Good example of a fractal
  - a fractal is a shape that will look almost, or even exactly, the same no matter what size it is viewed at.

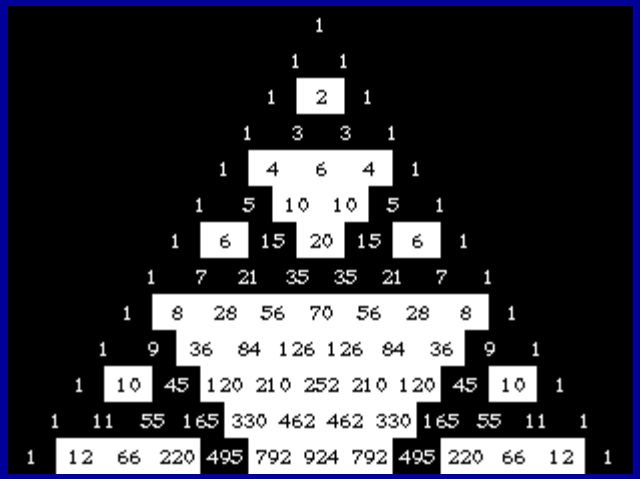
- Excellent use of recursion

Sierpinski triangle



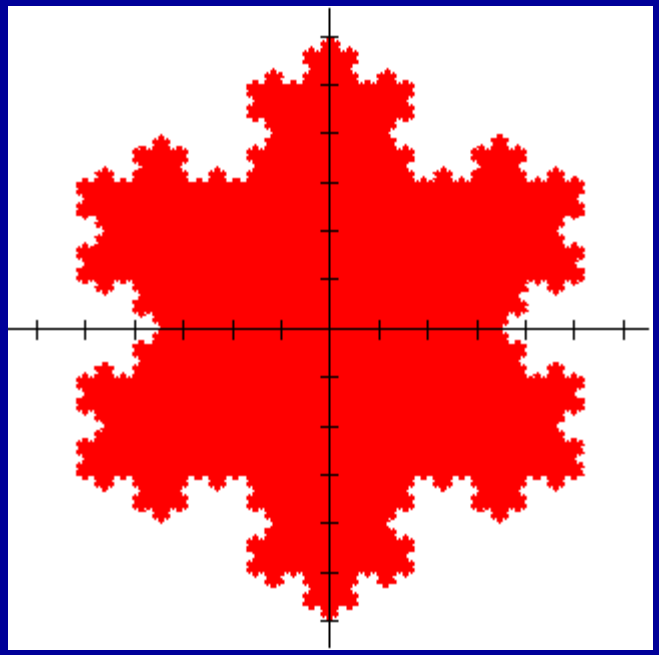
How to build it?

Pascal's triangle



Snow flakes …

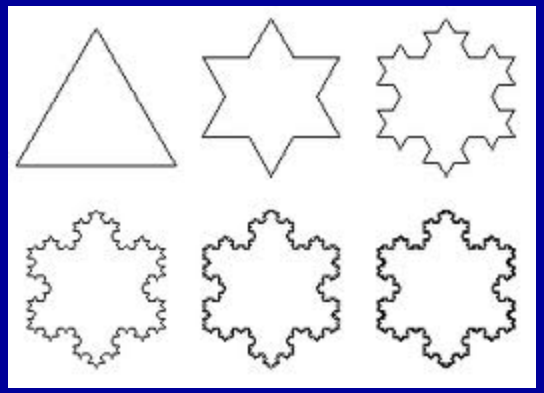
- A fractal, also known as the Koch island, which was first described by Helge von Koch in 1904



Building the snowflake

- How does it work?

- It is built by starting with an equilateral triangle, removing the inner third of each side, building another equilateral triangle at the location where the side was removed, and then repeating the process indefinitely.



Homework

- Homework #3 (linked lists) due tonight
- Quiz #3 (linked lists) tomorrow
- Homework #4 (recursion) due Tue 9/29

- Exam I on Thursday October 8

Readings

- Read as much as possible on recursion