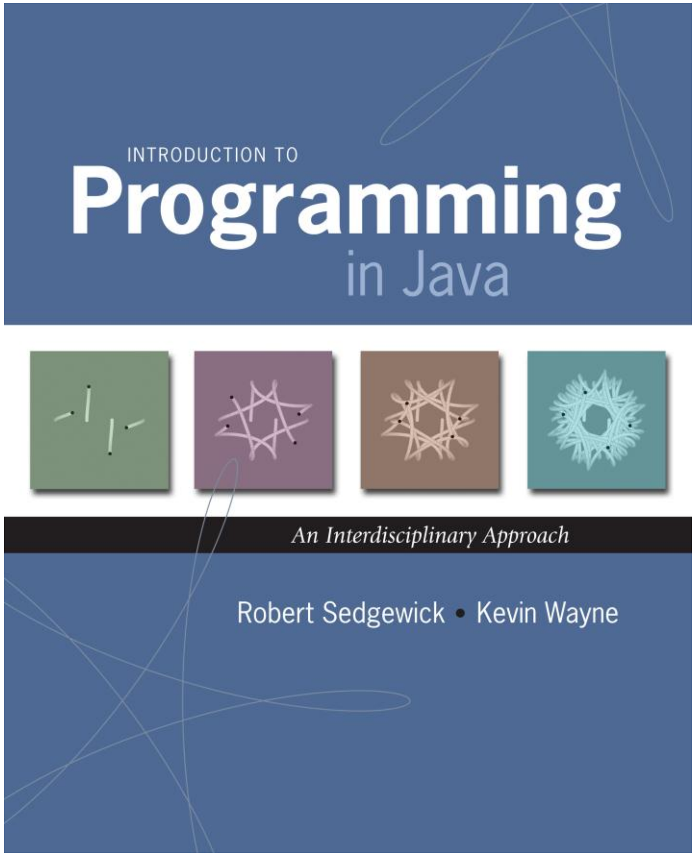
2.3 Recursion



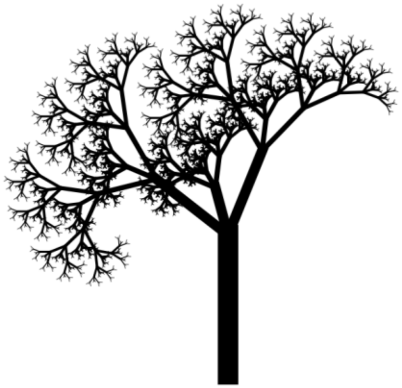
Overview

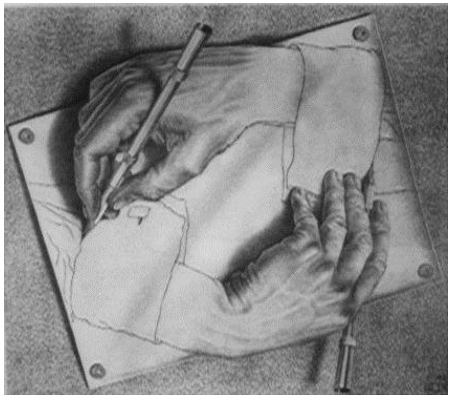
What is recursion? When one function calls itself directly or indirectly.

Why learn recursion?
- New mode of thinking.
- Powerful programming paradigm.

Many computations are naturally self-referential.
- Mergesort, FFT, gcd, depth-first search.
- Linked data structures.
- A folder contains files and other folders.

Closely related to mathematical induction.





Reproductive Parts
M. C. Escher, 1948

Greatest Common Divisor

Gcd. Find largest integer that evenly divides into p and q.

Ex. gcd(4032, 1272) = 24.

$$4032 = 2^6 \times 3^2 \times 7^1$$
$$1272 = 2^3 \times 3^1 \times 53^1$$
$$\text{gcd} = 2^3 \times 3^1 = 24$$

Applications.

- Simplify fractions: 1272/4032 = 53/168.
- RSA cryptosystem.

Greatest Common Divisor

Gcd. Find largest integer d that evenly divides into p and q.

Euclid's algorithm. [Euclid 300 BCE]

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q,\ p \,\%\, q) & \text{otherwise} \end{cases}$$

← base case

← reduction step, converges to base case

```
gcd(4032, 1272)  = gcd(1272, 216)
                 = gcd(216, 192)
                 = gcd(192, 24)
                 = gcd(24, 0)
                 = 24.
```

4032 = 3 × 1272 + 216

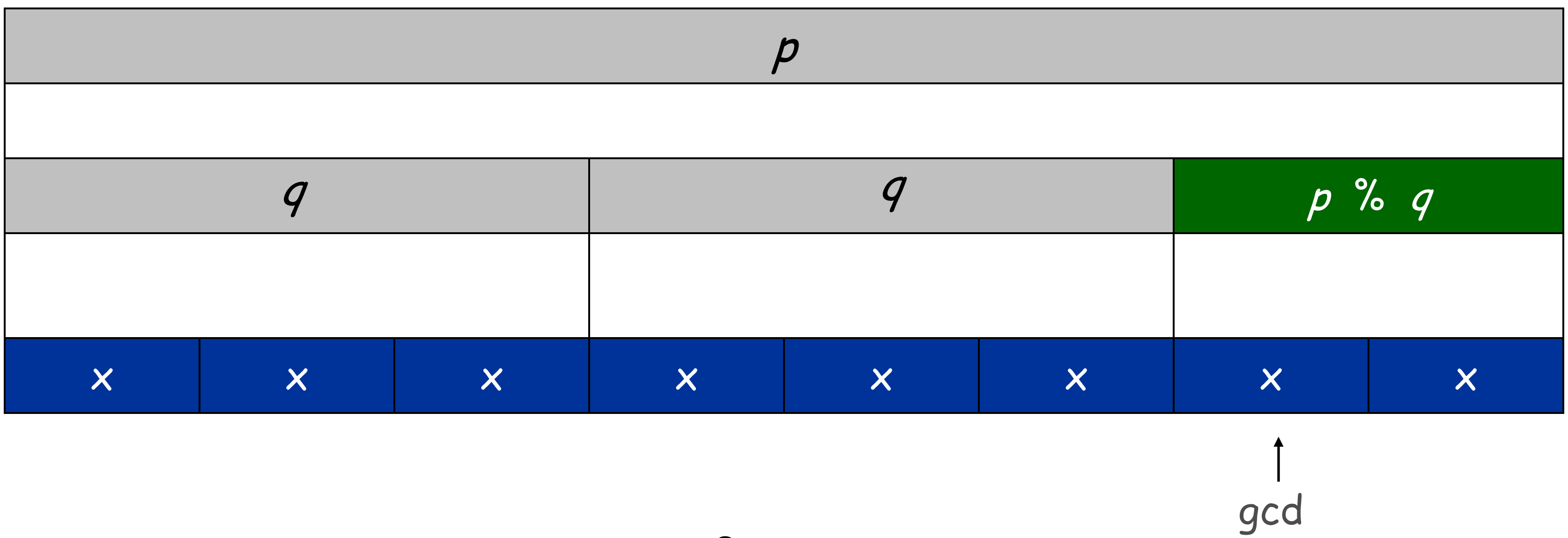
Greatest Common Divisor

Gcd. Find largest integer d that evenly divides into p and q.

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q,\ p \,\%\, q) & \text{otherwise} \end{cases}$$

← base case

← reduction step, converges to base case



p = 8x
q = 3x
gcd(p, q) = x

Greatest Common Divisor

Gcd. Find largest integer d that evenly divides into p and q.

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q, p \,\%\, q) & \text{otherwise} \end{cases}$$

← base case

← reduction step, converges to base case

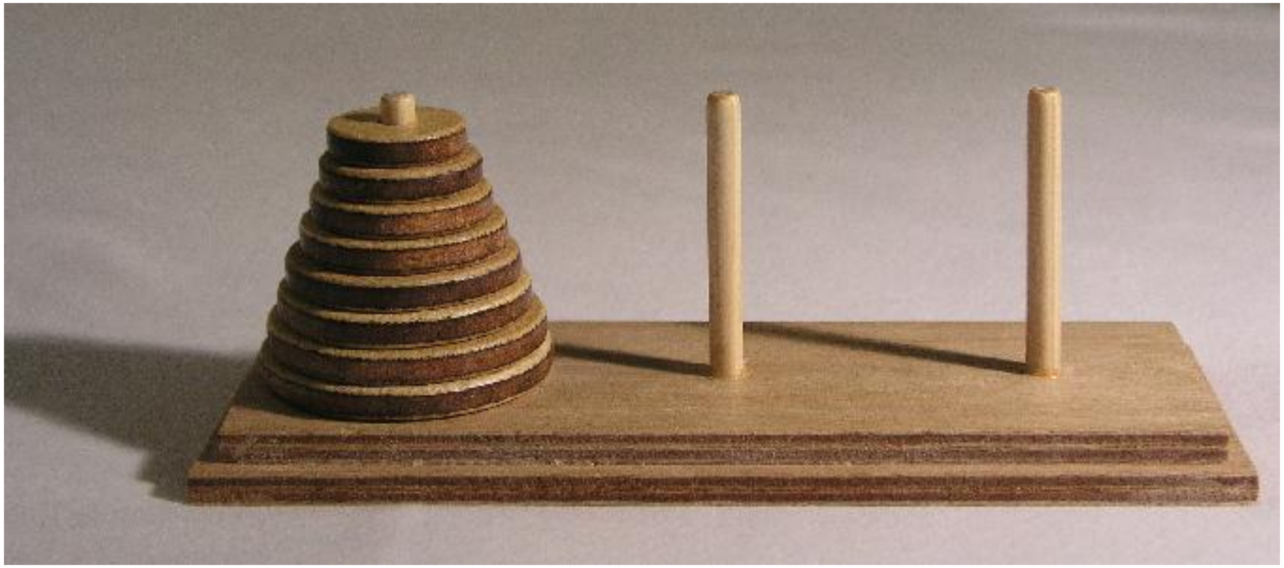
Java implementation.

```java
public static int gcd(int p, int q) {
    if (q == 0) return p;
    else return gcd(q, p % q);
}
```

← base case

← reduction step

Towers of Hanoi

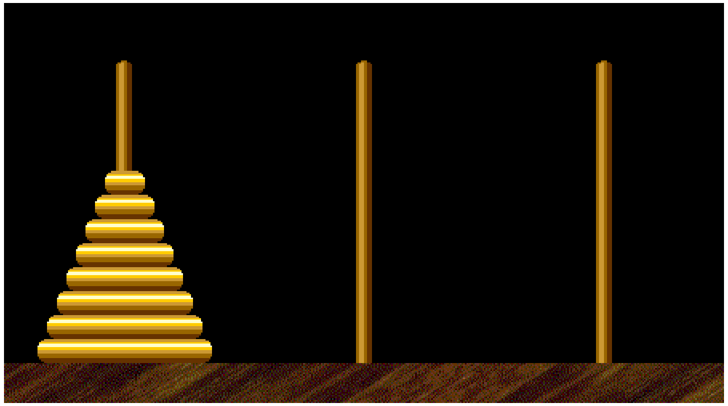


http://en.wikipedia.org/wiki/Image:Hanoiklein.jpg

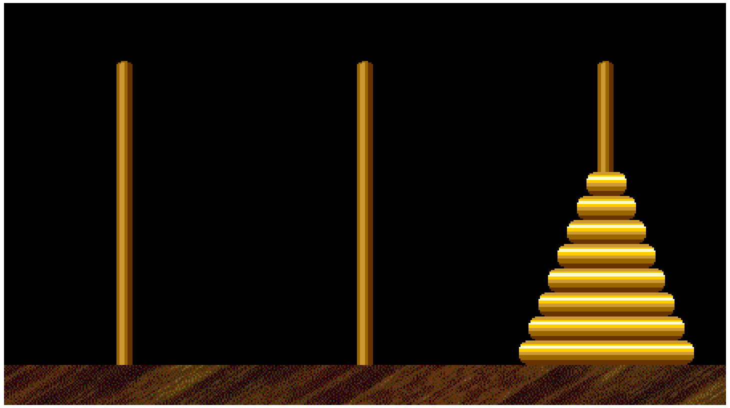
Towers of Hanoi

Move all the discs from the leftmost peg to the rightmost one.

- Only one disc may be moved at a time.
- A disc can be placed either on empty peg or on top of a larger disc.



start



finish

Edouard Lucas (1883)

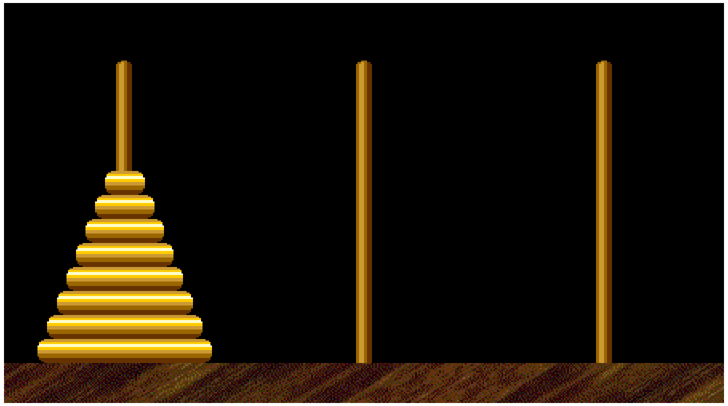
Towers of Hanoi Legend

Q. Is world going to end (according to legend)?

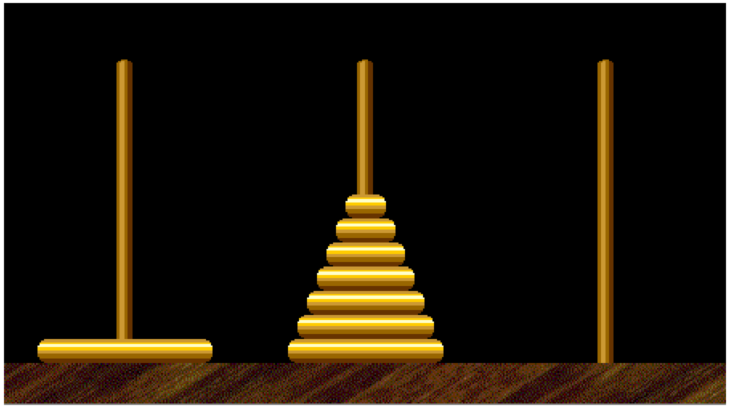
- 64 golden discs on 3 diamond pegs.
- World ends when certain group of monks accomplish task.

Q. Will computer algorithms help?

Towers of Hanoi: Recursive Solution

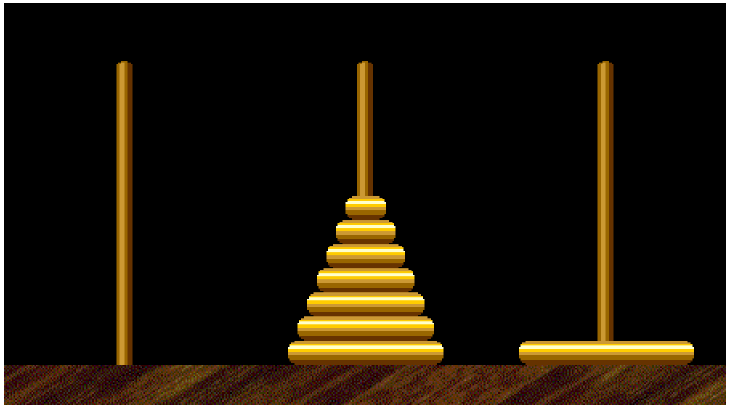


Move n-1 smallest discs right.

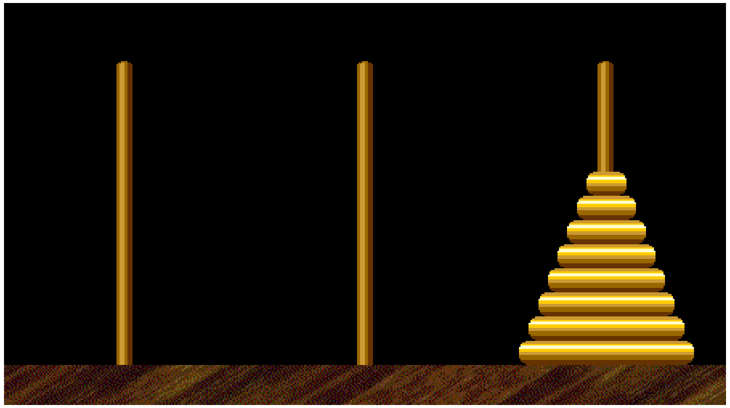


Move largest disc left.

cyclic wrap-around



Move n-1 smallest discs right.



Towers of Hanoi: Recursive Solution

```
public class TowersOfHanoi {

   public static void moves(int n, boolean left) {
      if (n == 0) return;
      moves(n-1, !left);
      if (left) System.out.println(n + " left");
      else      System.out.println(n + " right");
      moves(n-1, !left);
   }

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      moves(N, true);
   }

}
```

`moves(n, true)` : move discs 1 to n one pole to the left
`moves(n, false)`: move discs 1 to n one pole to the right

smallest disc

Towers of Hanoi: Recursive Solution

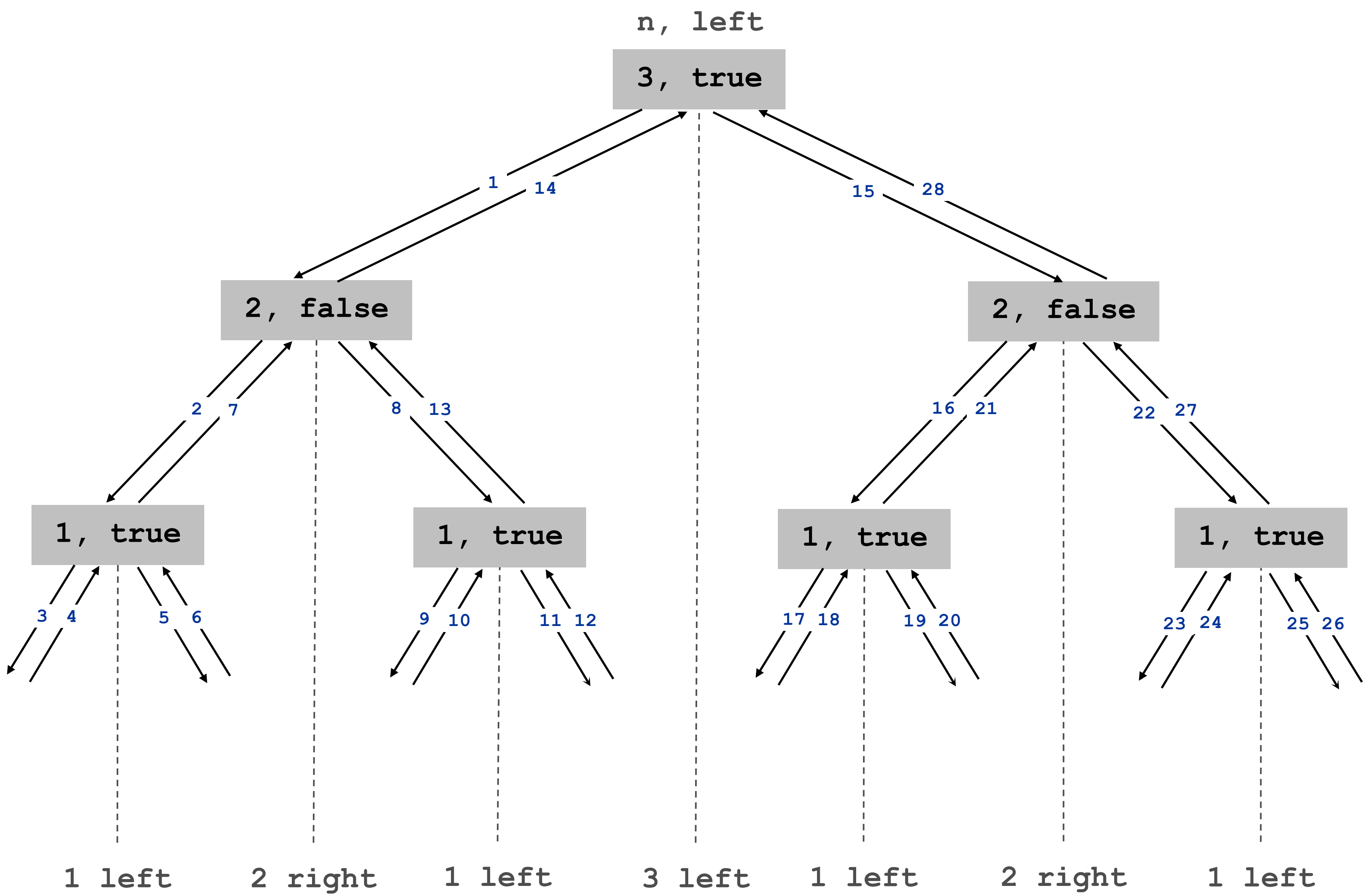
```
% java TowersOfHanoi 3
1 left
2 right
1 left
3 left
1 left
2 right
1 left
```

```
% java TowersOfHanoi 4
1 right
2 left
1 right
3 right
1 right
2 left
1 right
4 left
1 right
2 left
1 right
3 right
1 right
2 left
1 right
```

every other move is smallest disc

subdivisions of ruler

Towers of Hanoi: Recursion Tree



Towers of Hanoi: Properties of Solution

Remarkable properties of recursive solution.

- Takes 2^n - 1 moves to solve n disc problem.
- Sequence of discs is same as subdivisions of ruler.
- Every other move involves smallest disc.

Recursive algorithm yields non-recursive solution!

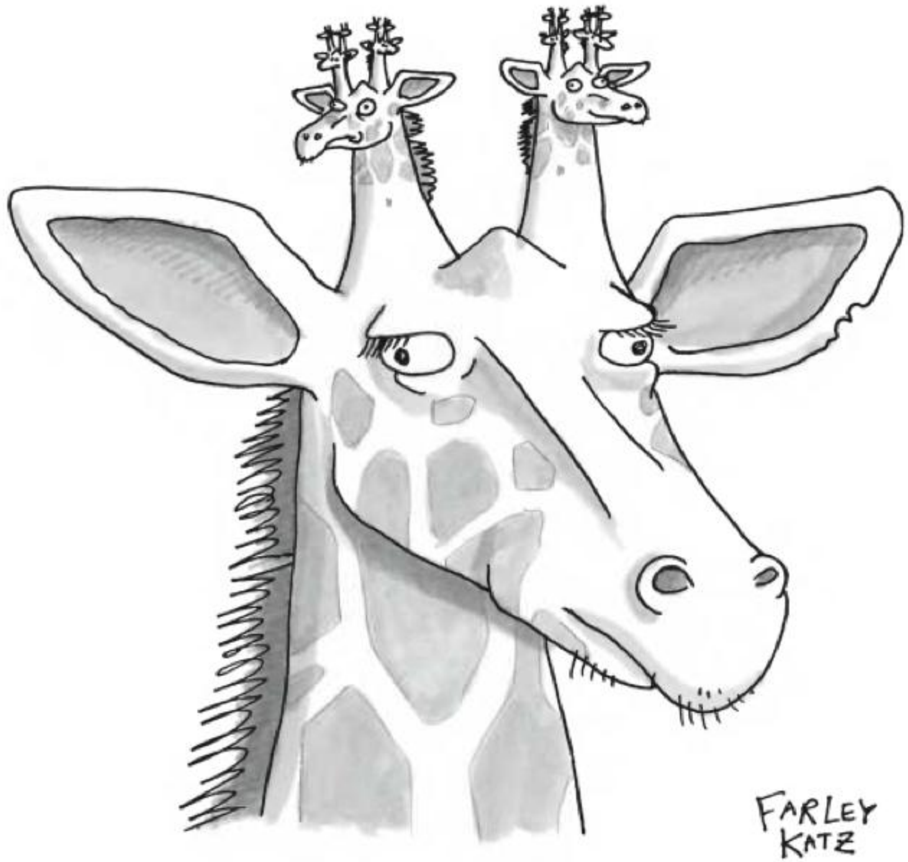
- Alternate between two moves:
  - move smallest disc to right if n is even
  - make only legal move not involving smallest disc

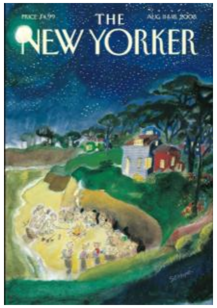
to left if n is odd

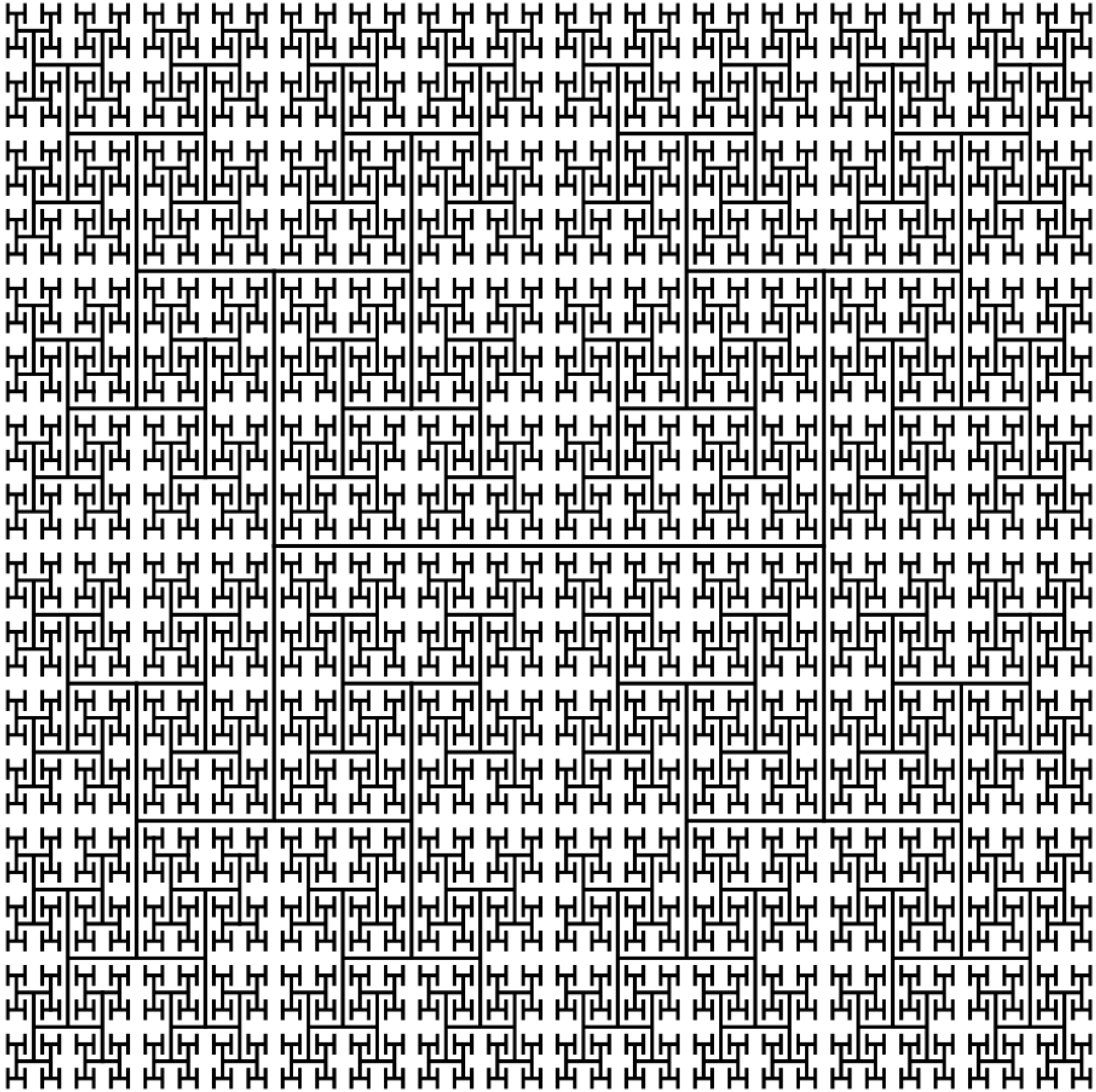
Recursive algorithm may reveal fate of world.

- Takes 585 billion years for n = 64 (at rate of 1 disc per second).
- Reassuring fact: any solution takes at least this long!

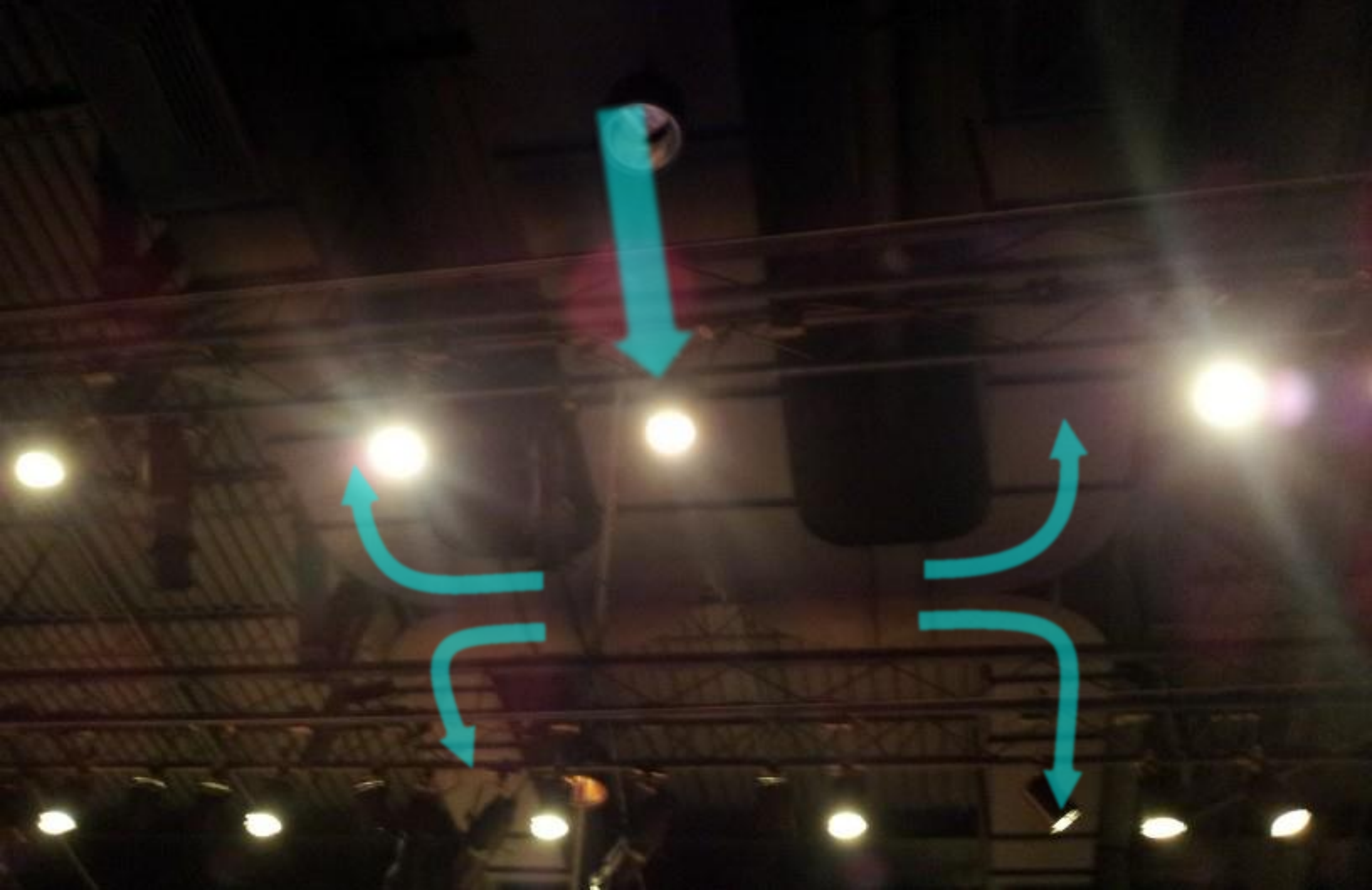
Recursive Graphics







Taplin Auditorium Ventilation System

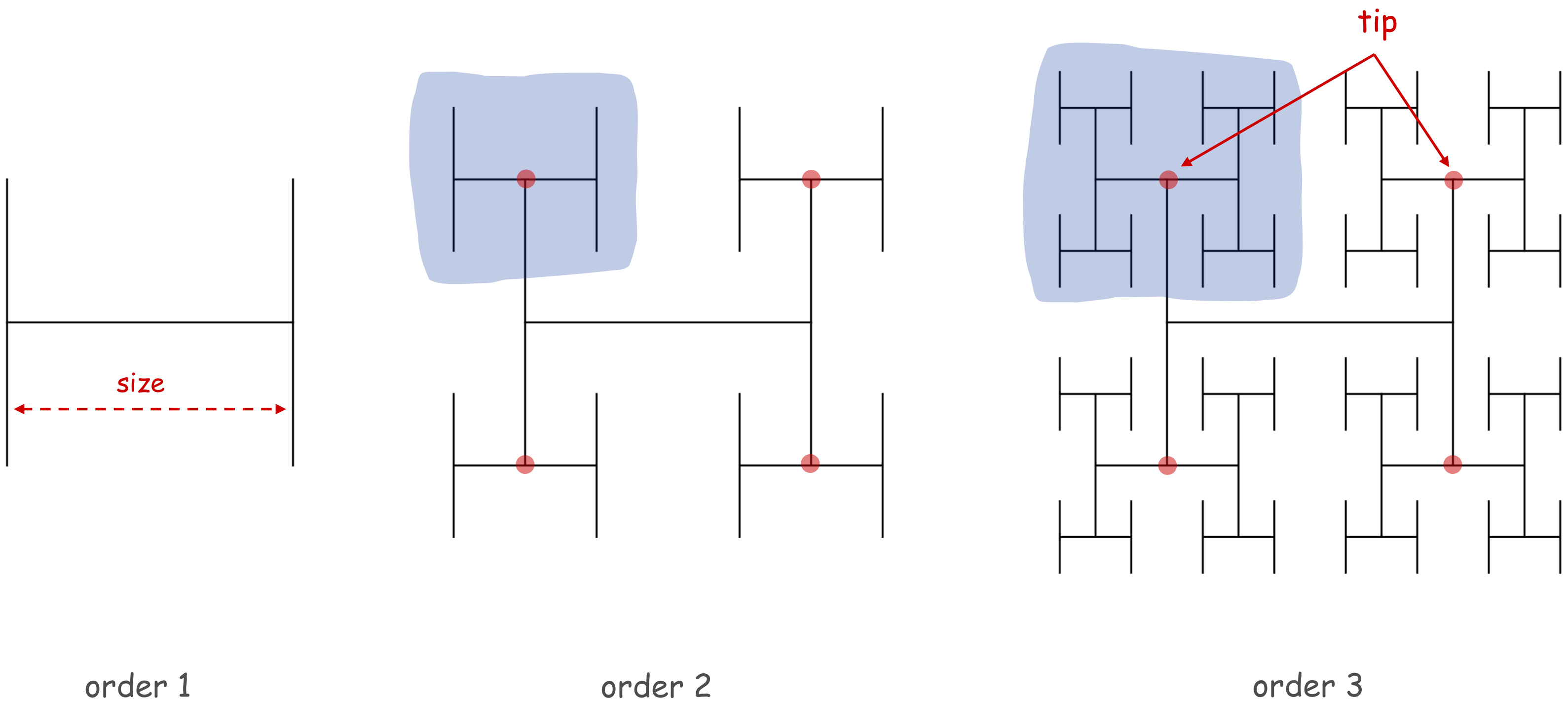


Htree

H-tree of order n.

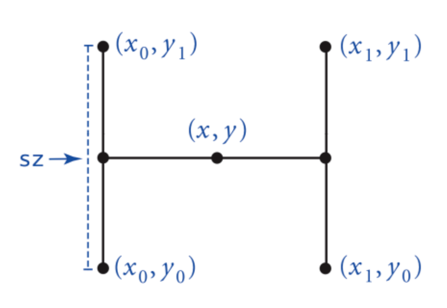
- Draw an H.
- Recursively draw 4 H-trees of order n-1, one connected to each tip.

and half the size



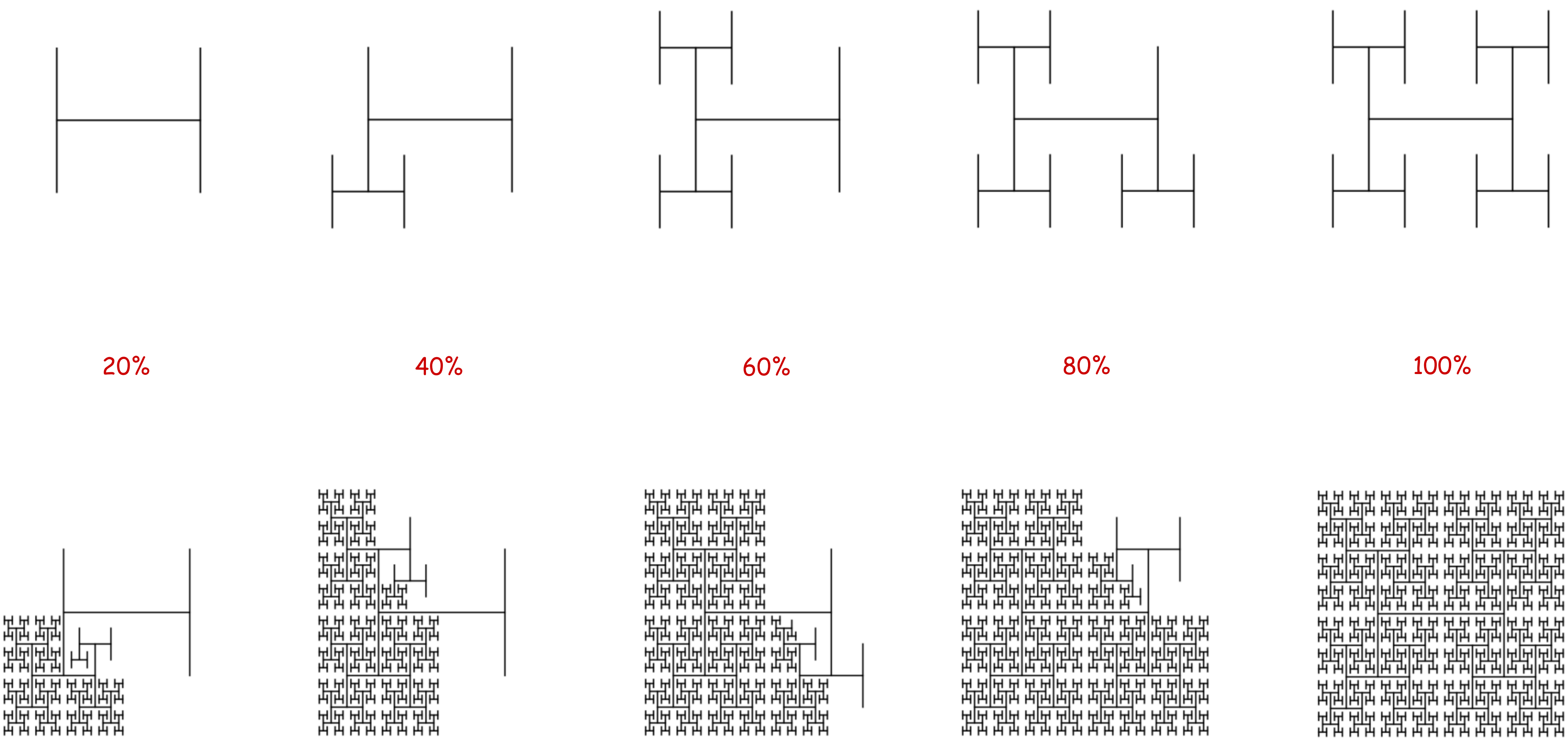
Htree in Java

```
public class Htree {
   public static void draw(int n, double sz, double x, double y) {
      if (n == 0) return;
      double x0 = x - sz/2, x1 = x + sz/2;
      double y0 = y - sz/2, y1 = y + sz/2;

      StdDraw.line(x0,  y, x1,  y);   ← draw the H, centered on (x, y)
      StdDraw.line(x0, y0, x0, y1);
      StdDraw.line(x1, y0, x1, y1);

      draw(n-1, sz/2, x0, y0);        ← recursively draw 4 half-size Hs
      draw(n-1, sz/2, x0, y1);
      draw(n-1, sz/2, x1, y0);
      draw(n-1, sz/2, x1, y1);
   }

   public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);
      draw(n, .5, .5, .5);
   }
}
```



Animated H-tree

Animated H-tree. Pause for 1 second after drawing each H.

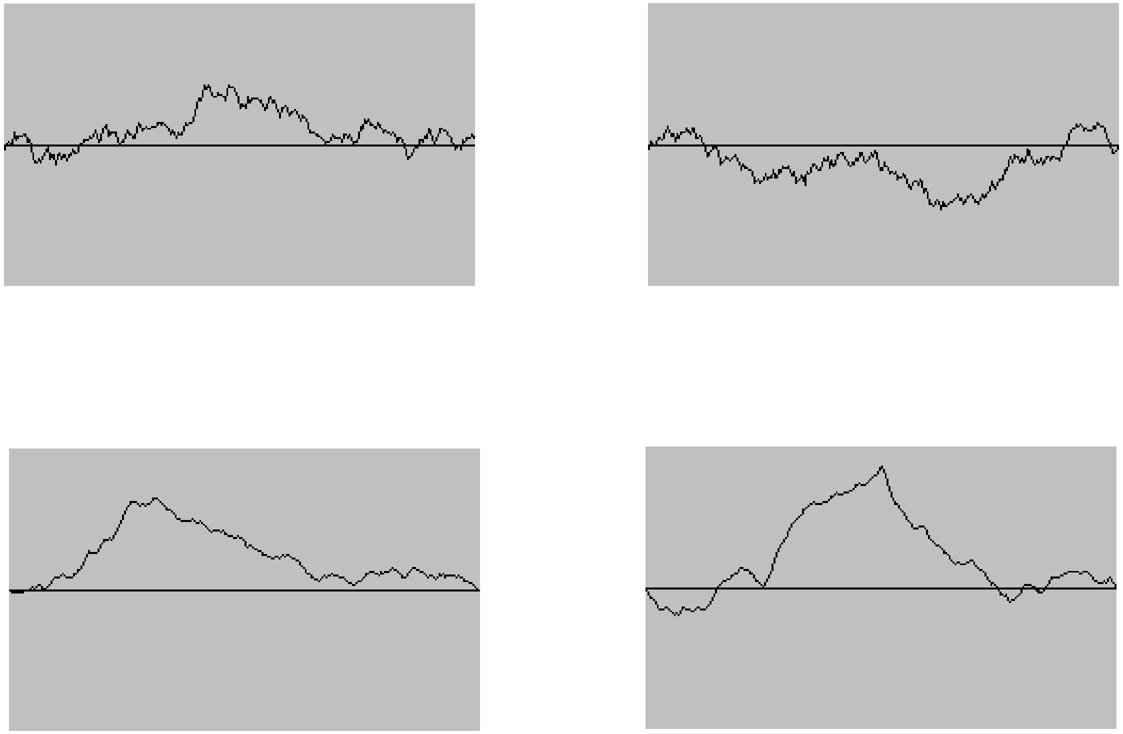


Fractional Brownian Motion

Fractional Brownian Motion

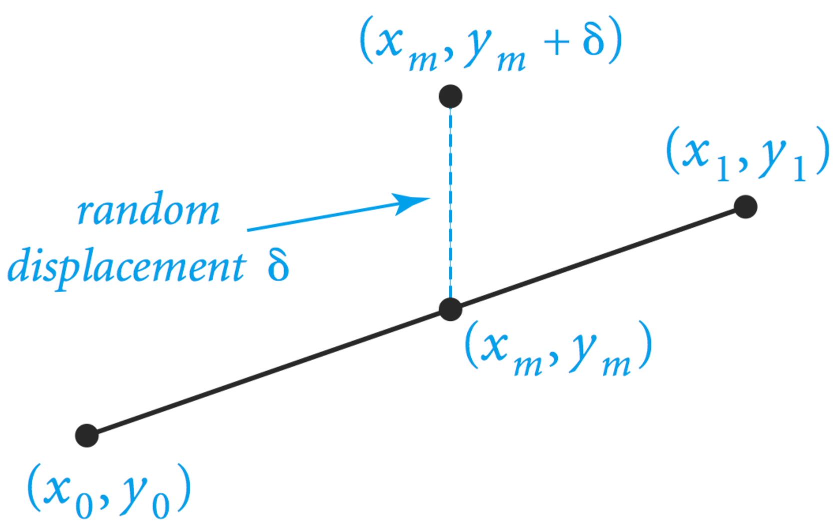
Physical process which models many natural and artificial phenomenon.

- Price of stocks.
- Dispersion of ink flowing in water.
- Rugged shapes of mountains and clouds.
- Fractal landscapes and textures for computer graphics.



Midpoint displacement method.

- Maintain an interval with endpoints (x_0, y_0) and (x_1, y_1).
- Divide the interval in half.
- Choose δ at random from Gaussian distribution.
- Set $x_m = (x_0 + x_1)/2$ and $y_m = (y_0 + y_1)/2 + \delta$.
- Recur on the left and right intervals.



Simulating Brownian Motion: Java Implementation

Midpoint displacement method.

- Maintain an interval with endpoints (x_0, y_0) and (x_1, y_1).
- Divide the interval in half.
- Choose δ at random from Gaussian distribution.
- Set $x_m = (x_0 + x_1)/2$ and $y_m = (y_0 + y_1)/2 + \delta$.
- Recur on the left and right intervals.

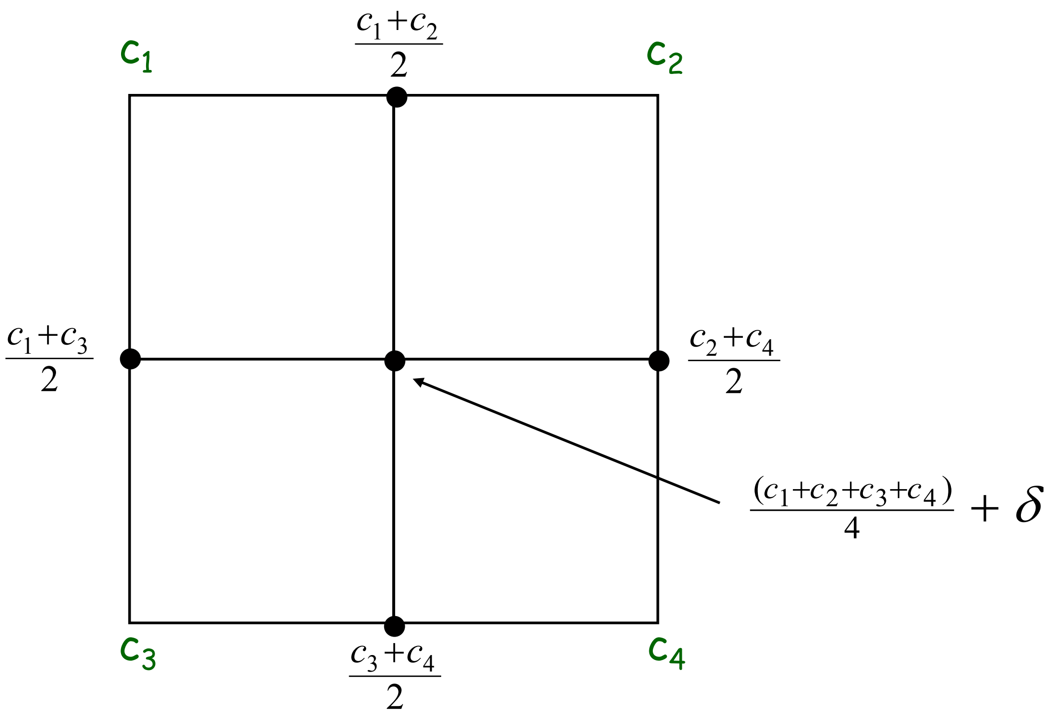
```
public static void curve(double x0, double y0,
                         double x1, double y1, double var) {
   if (x1 - x0 < 0.01) {
      StdDraw.line(x0, y0, x1, y1);
      return;
   }
   double xm = (x0 + x1) / 2;
   double ym = (y0 + y1) / 2;
   ym += StdRandom.gaussian(0, Math.sqrt(var));
   curve(x0, y0, xm, ym, var/2);
   curve(xm, ym, x1, y1, var/2);
}
```

variance halves at each level;
change factor to get different shapes

Plasma Cloud

Plasma cloud centered at (x, y) of size s.

- Each corner labeled with some grayscale value.
- Divide square into four quadrants.
- The grayscale of each new corner is the average of others.
  - center: average of the four corners + random displacement
  - others: average of two original corners
- Recur on the four quadrants.



Plasma Cloud

Brownian Landscape

Reference: http://www.geocities.com/aaron_torpy/gallery.htm

Brown

Robert Brown (1773-1858)