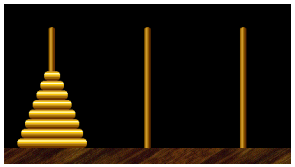
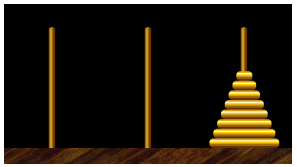
2.3 Recursion



start



finish

Overview

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

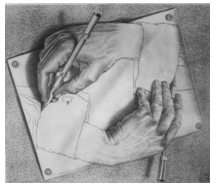
Why learn recursion?

- New mode of thinking.
- Powerful programming tool.
- Divide-and-conquer paradigm.

Many computations are naturally self-referential.

- Quicksort, FFT, gcd.
- Linked data structures.
- A directory contains files and other directories.

Closely related to mathematical induction.



Drawing Hands
M. C. Escher, 1948

Greatest Common Divisor

Gcd. Find largest integer d 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 (stay tuned).

Greatest Common Divisor

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

Euclid's algorithm.

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

← base case

← reduction step, converges to base case

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

Euclid, 300 BCE

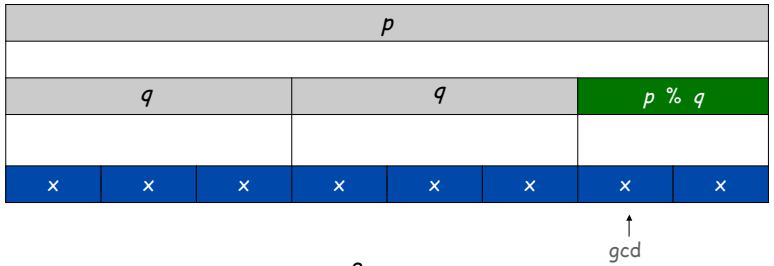
Greatest Common Divisor

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.

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

← base case

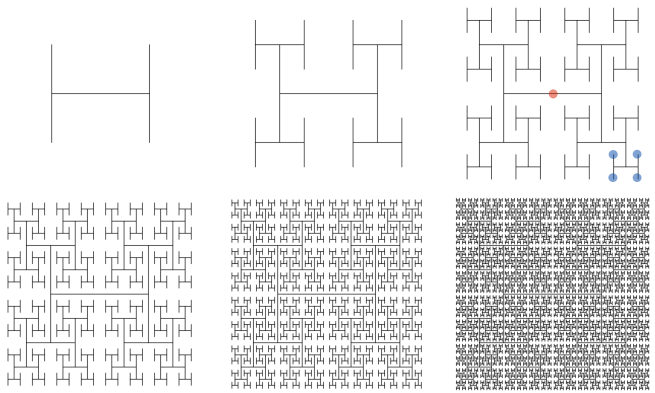
← reduction step

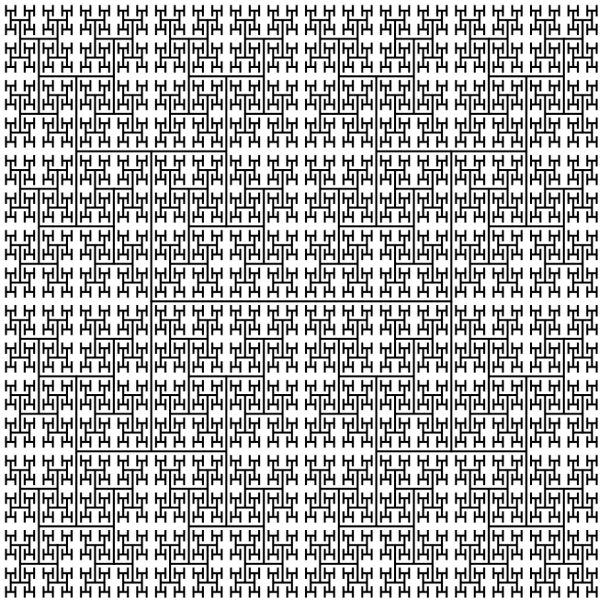
Htree

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 ↙



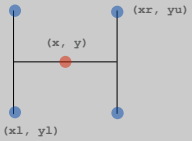


Htree in Java

```
public class Htree {
   public static void draw(int n, double sz, double x, double y) {
      if (n == 0) return;
      double xl = x - sz/2, xr = x + sz/2;
      double yl = y - sz/2, yu = y + sz/2;

      StdDraw.line(xl,  y, xr,  y);
      StdDraw.line(xl, yl, xl, yu);
      StdDraw.line(xr, yl, xr, yu);

      draw(n-1, sz/2, xl, yl);
      draw(n-1, sz/2, xl, yu);
      draw(n-1, sz/2, xr, yl);
      draw(n-1, sz/2, xr, yu);
   }

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

draw the H, centered on (x, y)

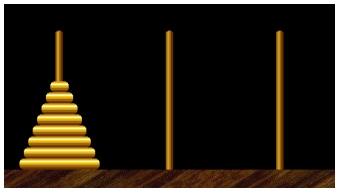
recursively draw 4 half-size Hs



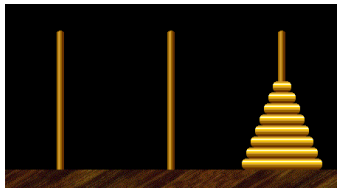
Towers of Hanoi

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

Towers of Hanoi demo

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

```
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

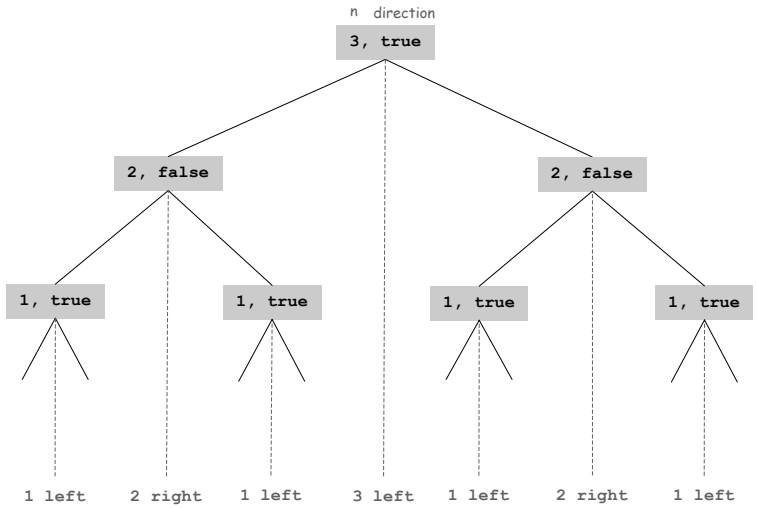
Towers of Hanoi: Recursive Solution

```
% 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
```

subdivisions of ruler

Towers of Hanoi: Recursion Tree



Properties of Towers of Hanoi Solution

Remarkable properties of recursive solution.

- Takes $2^n - 1$ steps to solve n disc problem.
- Sequence of discs is same as subdivisions of ruler.
- Smallest disc always moves in same direction.

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!

Divide-and-Conquer

Divide-and-conquer paradigm.

- Break up problem into smaller subproblems of same structure.
- Solve subproblems recursively using same method.
- Combine results to produce solution to original problem.

Divide et impera. Veni, vidi, vici. - Julius Caesar

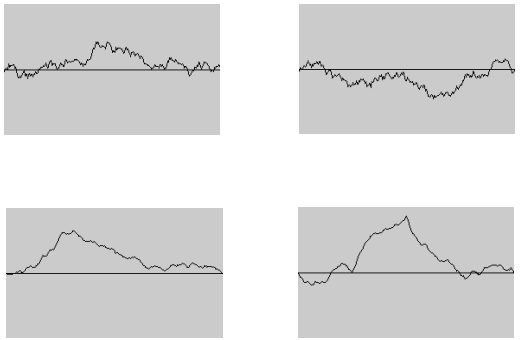
Many important problems succumb to divide-and-conquer.

- Quicksort for sorting.
- FFT for signal processing.
- Multigrid methods for solving PDEs.
- Adaptive quadrature for integration.
- Hilbert curve for domain decomposition.
- Integer arithmetic for RSA cryptography.
- Quad-tree for efficient N-body simulation.
- Midpoint displacement method for Brownian motion.

Fractional Brownian Motion

Physical process which models many natural and artificial phenomenon.

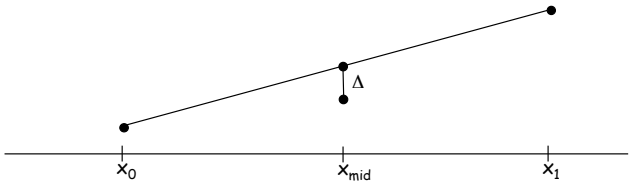
- Dispersion of ink flowing in water.
- Price of stocks.
- Rugged shapes of mountains and clouds.
- Fractal landscapes and textures for computer graphics.



Simulating Brownian Motion

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 $x_{mid} = (x_0 + x_1)/2$ and $y_{mid} = (y_0 + y_1)/2 + \Delta$.
- Recur on the left and right intervals.



Simulating Brownian Motion in Java

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 $x_{mid} = (x_0 + x_1)/2$ and $y_{mid} = (y_0 + y_1)/2 + \Delta$.
- Recur on the left and right intervals.

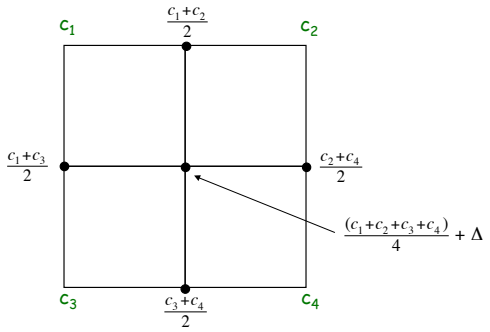
```
public static void curve(double x0, double y0,
                         double x1, double y1, double var) {
   if (x1 - x0 < 0.01) {
      StdDraw.line(x0, y0, x1, y1);
      return;
   }
   double xm = (x0 + x1) / 2;
   double ym = (y0 + y1) / 2;
   ym += StdRandom.gaussian(0, Math.sqrt(var));
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

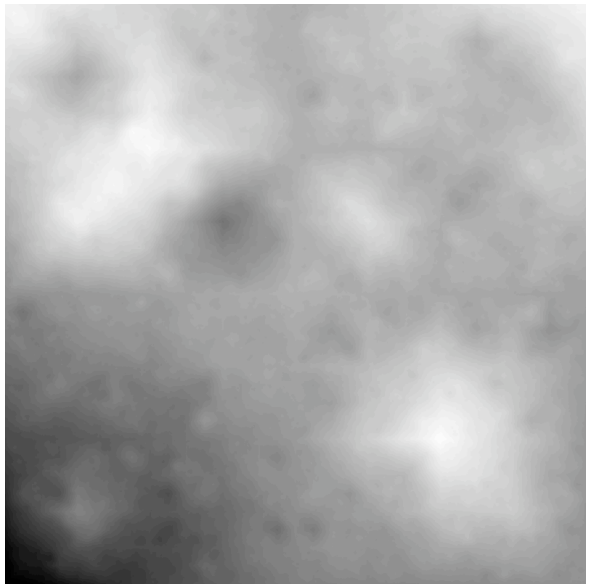
Plasma Cloud

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.



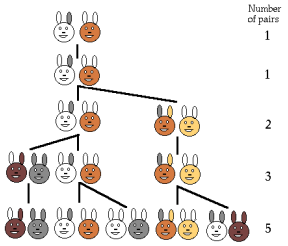
Plasma Cloud (Grayscale)



Fibonacci Numbers

Fibonacci numbers. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$



Fibonacci rabbits

L. P. Fibonacci
(1170 - 1250)

Possible Pitfalls With Recursion

Fibonacci numbers. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

A natural for recursion?

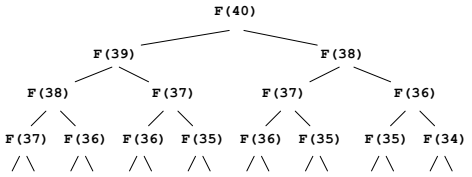
```
public static int F(int n) {
   if (n == 0 || n == 1) return n;
   else return F(n-1) + F(n-2);
}
```

Spectacularly inefficient Fibonacci

Observation. It takes a really long time to compute `F(40)`.

Possible Pitfalls With Recursion

Caveat. Can easily write remarkably inefficient programs.



recursion tree for naïve Fibonacci function

F(39) is computed once.
F(38) is computed 2 times.
F(37) is computed 3 times.
F(36) is computed 5 times.
F(35) is computed 8 times.
…
F(0) is computed 165,580,141 times.

331,160,281 function calls for F(40).

Binet's formula. $F(n) = \dfrac{\phi^n - (1-\phi)^n}{\sqrt{5}} = \lfloor \phi^n / \sqrt{5} \rfloor$

$\phi = 1.61803398875$

$\phi^2 = \phi + 1$

Summary

How to write simple recursive programs?

- Base case, reduction step.
- Trace the execution of a recursive program.
- Use pictures.

Why learn recursion?

- New mode of thinking.
- Powerful programming tool.

Many important problems have elegant divide-and-conquer solutions.