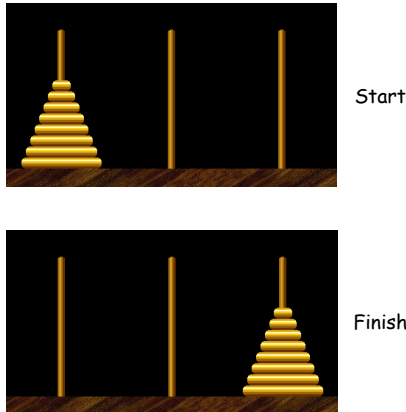


2.7: Recursion



Introduction to Computer Science · Robert Sedgewick and Kevin Wayne · <http://www.cs.Princeton.EDU/IntroCS>

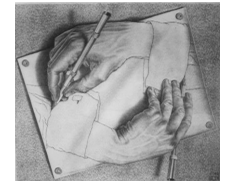
What is recursion? When one function calls *itself* directly or indirectly.

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.



Drawing Hands
M. C. Escher, 1948

Closely related to mathematical induction.

Greatest Common Divisor

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

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

- ← base case
- ← reduction step, converges to base case

$$\begin{aligned} \text{gcd}(4032, 1272) &= \text{gcd}(1272, 216) \\ &= \text{gcd}(216, 192) \\ &= \text{gcd}(192, 24) \\ &= \text{gcd}(24, 0) \\ &= 24. \end{aligned}$$

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



Euclid, 300 BCE

Applications.

- Simplify fractions: $1272/4032 = 53/168$.
- RSA cryptosystem (stay tuned).
- History of algorithms.

Greatest Common Divisor

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

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

- ← base case
- ← reduction step, converges to base case

| | | | | | | | | |
|---|---|---|---|---|---|-------|---|--|
| p | | | | | | | | |
| q | | | q | | | p % q | | |
| x | x | x | x | x | x | x | x | |

$$\begin{aligned} p &= 8x \\ q &= 3x \\ \text{gcd}(p, q) &= x \end{aligned}$$

↑
gcd

Greatest Common Divisor

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

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

- ← 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);
}
```

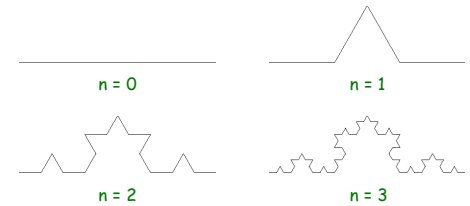
- ← base case
- ← reduction step



Koch Snowflake

Koch curve of order n .

- Draw curve of order $n-1$.
- Turn 60° .
- Draw curve of order $n-1$.
- Turn -120° .
- Draw curve of order $n-1$.
- Turn 60° .
- Draw curve of order $n-1$.

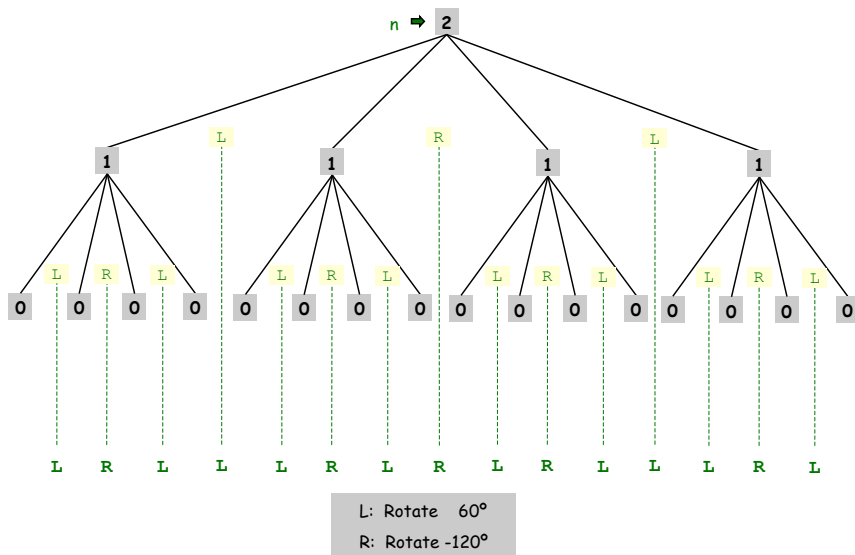


```
public static void koch(int n, double size) {
    if (n == 0) StdDraw.goForward(size);
    else {
        koch(n-1, size);
        StdDraw.rotate(60);
        koch(n-1, size);
        StdDraw.rotate(-120);
        koch(n-1, size);
        StdDraw.rotate(60);
        koch(n-1, size);
    }
}
```

5

6

Koch Curve: Recursion Tree

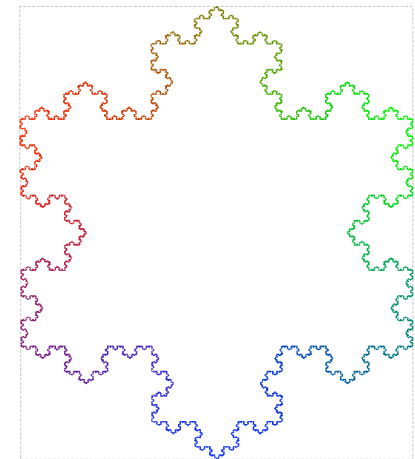


7

Koch Snowflake

Koch snowflake of order n .

- Draw Koch curve of order n .
- Turn -120° .
- Draw Koch curve of order n .
- Turn -120° .
- Draw Koch curve of order n .



10

Koch Snowflake in Java

```

public class Koch {
    public static void koch(int n, double size) { } ← just did this

    public static void main(String args[]) {
        int N = Integer.parseInt(args[0]); ← compute parameters
        int width = 512;
        int height = (int) (2 * width / Math.sqrt(3));
        double size = width / Math.pow(3.0, N);

        StdDraw.create(width, height);
        StdDraw.go(0, width * Math.sqrt(3) / 2);
        StdDraw.penDown();
        koch(N, size);
        StdDraw.rotate(-120);
        koch(N, size);
        StdDraw.rotate(-120);
        koch(N, size);
        StdDraw.show(); ← draw it
    }
}

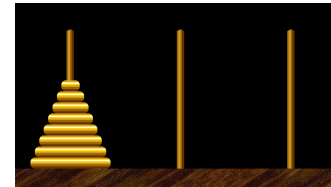
```

11

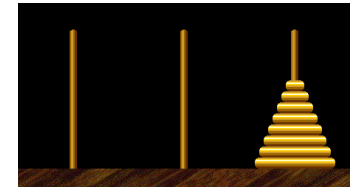
Towers of Hanoi

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

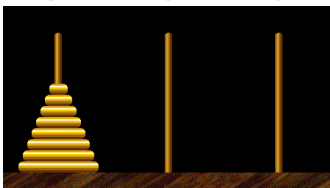


Edouard Lucas (1883)

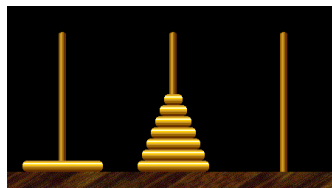
12

Towers of Hanoi: Recursive Solution

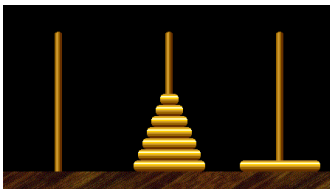
A B C



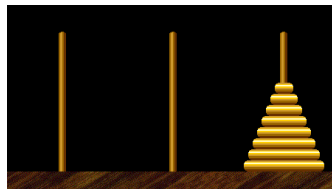
Move N-1 smallest discs to pole B.



Move largest disc to pole C.



Move N-1 smallest discs to pole C.



14

Towers of Hanoi Legend

Is world going to end (according to legend)?

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

Q. Will computer algorithms help?

15

Towers of Hanoi: Recursive Solution

```
public class Hanoi {

    public static void hanoi(int n, String from, String temp,
        String to) {

        if (n == 0) return;
        hanoi(n-1, from, to, temp);
        System.out.println("Move disc " + n +
            " from " + from + " to " + to);
        hanoi(n-1, temp, from, to);
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        hanoi(N, "A", "B", "C");
    }
}
```

Towers of Hanoi: Recursive Solution

```
% java Hanoi 3
Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
Move disc 1 from A to C
```

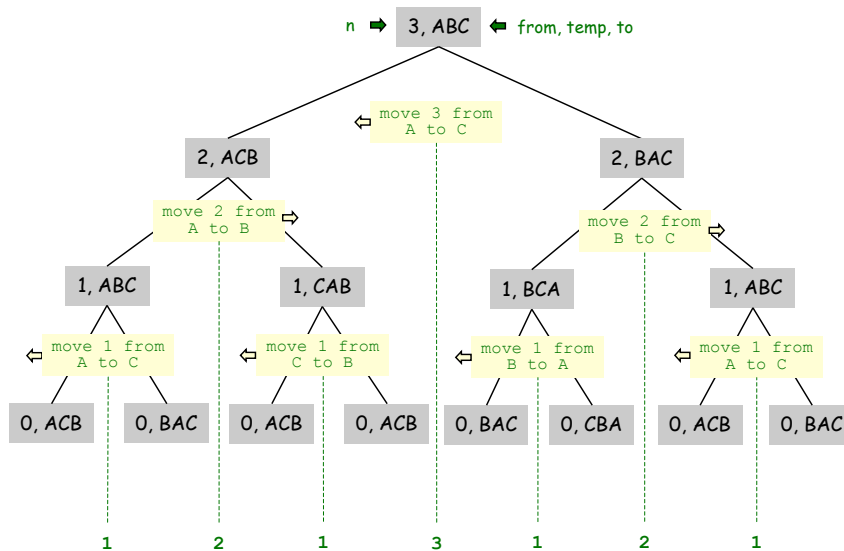
```
% java Hanoi 4
Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
Move disc 3 from A to B
Move disc 1 from C to A
Move disc 2 from C to B
Move disc 1 from A to B
Move disc 4 from A to C
Move disc 1 from B to C
Move disc 2 from B to A
Move disc 1 from C to A
Move disc 3 from B to C
Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
```

↑
subdivisions of ruler

16

17

Towers of Hanoi: Recursion Tree



18

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 (left) if N is even (odd)
 - make only legal move not involving smallest disc

Recursive algorithm may reveal fate of world.

- Takes 348 centuries for $N = 40$, assuming rate of 1 disc per second.
- Reassuring fact: any solution takes at least this long!

19

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.

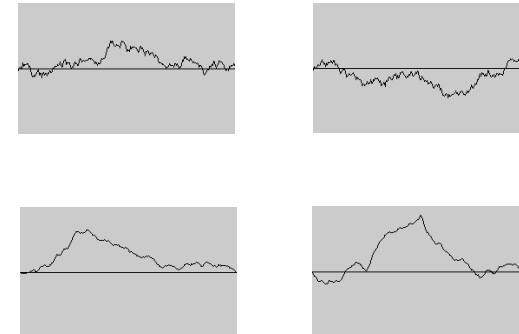
- Quicksort for sorting.
- FFT for signal processing.
- Multigrid methods for solving PDEs.
- Adaptive quadrature for integration.
- Hilbert curve for domain decomposition.
- Integer arithmetic for RSA cryptography.
- Quad-tree for efficient N-body simulation.
- Midpoint displacement method for Brownian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

20

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.

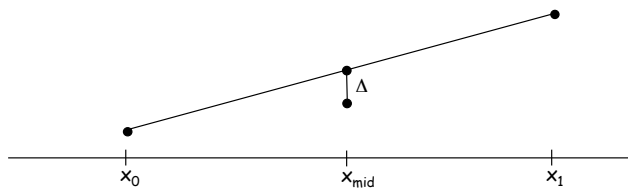


21

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



22

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

```
public static void midpoint(double x0, double y0,
                           double x1, double y1, double var) {
    if (x1 - x0 < 1.0) return;
    double displacement = MyMath.gaussian(0, Math.sqrt(var));
    double xmid = 0.5 * (x0 + x1);
    double ymid = 0.5 * (y0 + y1) + displacement;
    midpoint(x0, y0, xmid, ymid, var/2);
    StdDraw.go(xmid, ymid);
    midpoint(xmid, ymid, x1, y1, var/2);
}
```

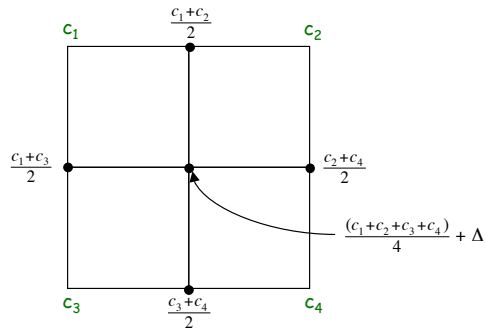
variance halves at each level

23

Plasma Cloud

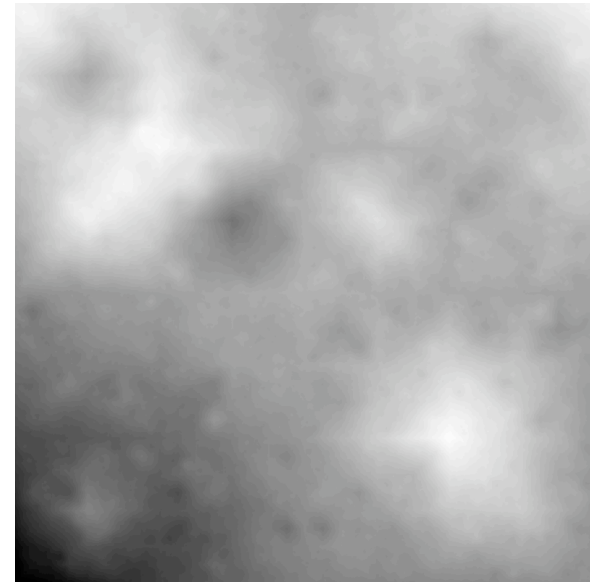
Plasma cloud centered at (x, y) of size s .

- 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.



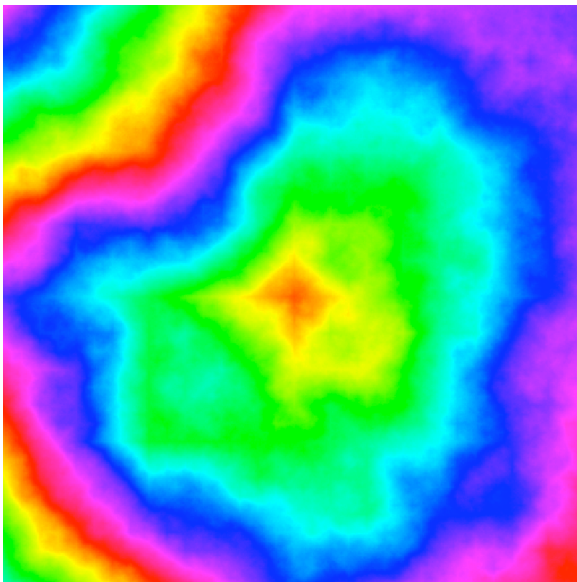
24

Plasma Cloud (Grayscale)



25

Plasma Cloud (Color)



26

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.



Towers of Hanoi by W. A. Schloss.

28