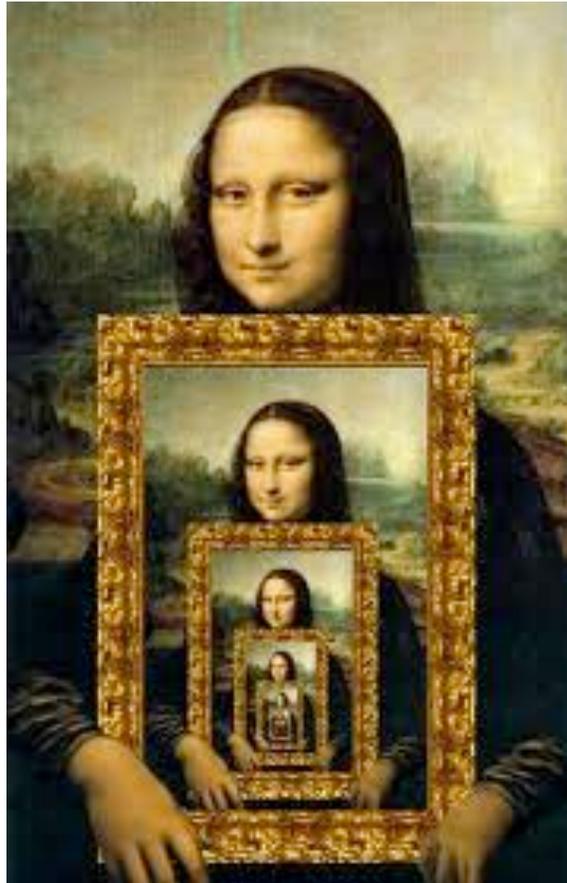




2.3 Recursion



Overview

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

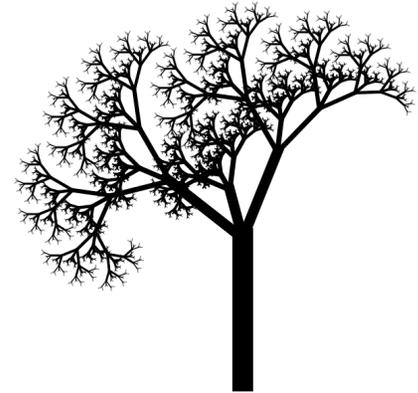
Why learn recursion?

- New mode of thinking.
- Powerful programming paradigm.

Many computations are naturally self-referential.

- Binary search, mergesort, FFT, **GCD**.
- Linked data structures.
- A folder contains files and other folders.

Closely related to mathematical induction.



M. C. Escher, 1956

Mathematical Induction

Mathematical induction. Prove a statement involving an integer N by

- **base case:** Prove it for some specific N (usually 0 or 1).
- **induction step:** Assume it to be true for all positive integers less than N , use that fact to prove it for N .

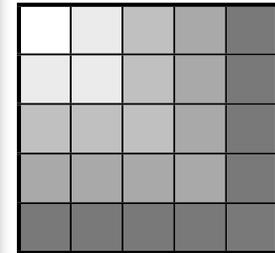
Ex. Sum of the first N odd integers is N^2 .

Base case: True for $N = 1$.

Induction step:

- Let $T(N)$ be the sum of the first N odd integers: $1 + 3 + 5 + \dots + (2N - 1)$.
- Assume that $T(N-1) = (N-1)^2$.
- $T(N) = T(N-1) + (2N - 1)$
 $= (N-1)^2 + (2N - 1)$
 $= N^2 - 2N + 1 + (2N - 1)$
 $= N^2$

$$\begin{aligned} 1 &= 1 \\ 1 + 3 &= 4 \\ 1 + 3 + 5 &= 9 \\ 1 + 3 + 5 + 7 &= 16 \\ 1 + 3 + 5 + 7 + 9 &= 25 \\ &\dots \end{aligned}$$



Recursive Program

Recursive Program. Implement a function having integer arguments by

- **base case:** Do something specific in response to "base" argument values.
- **reduction step:** Assume the function works for the base case, and use the function to implement **itself** for general argument values.

```
public static String convert(int x)
{
    if (x == 1) return "1";
    return convert(x/2) + (x % 2);
}
```

automatic cast to
String
(either "0" or "1")

Ex 1. Convert positive `int` to binary string.

Base case: return "1" for $x = 1$.

Reduction step:

- convert $x/2$ to binary
- append "0" if x even
- append "1" if x odd

37 18
"100101" = "10010" + "1"

Recursive Program

Recursive Program. Implement a function having integer arguments by

- **base case:** Implementing it for some specific values of the arguments.
- **reduction step:** Assume the function works for smaller values of its arguments and use it to implement it for the given values.

```
public class Binary
{
    public static int convert(int x)
    {
        if (x == 1) return "1";
        else return convert(x/2) + (x % 2);
    }

    public static void main(String[] args)
    {
        int x = Integer.parseInt(args[0]);
        System.out.println(convert(x));
    }
}
```

```
% java Binary 6
110
% java Binary 37
100101
% java Binary 999999
11110100001000111111
```

Use **mathematical induction** to prove recursive programs correct

Recursive Program. Implement a function having integer arguments by

- base case: Implementing it for some specific values of the arguments.
- reduction step: Assume the function works for smaller values of its arguments and use it to implement it for the given values.

Mathematical induction. Prove a statement involving an integer N by

- base case: Prove it for some specific N (usually 0 or 1).
- induction step: Assume it to be true for all positive integers less than N , use that fact to prove it for N .

Ex. Proof by induction for `convert()`.

Base case: returns "1" for $x = 1$.

Induction step:

- Assume correct for $x/2$.
- correct to append "0" if x even since $x = 2(x/2)$
- correct to append "1" if x odd since $x = 2(x/2) + 1$

```
public static String convert(int x)
{
    if (x == 1) return "";
    return convert(x/2) + (x % 2);
}
```

Recursion vs. Iteration

Every program with 1 recursive call corresponds to a loop.

```
public static String convert(int x)
{
    if (x == 1) return "1";
    return convert(x/2) + (x % 2);
}
```

```
public static String convertNR(int x)
{
    String s = "1";
    while (x > 1)
    {
        s = (x % 2) + s;
        x = x/2;
    }
    return s;
}
```

Reasons to use recursion:

- code more compact
- easier to understand
- easier to reason about correctness
- easy to add multiple recursive calls (stay tuned)

Reasons **not** to use recursion: (stay tuned)

Greatest Common Divisor

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

Ex. $\text{gcd}(4032, 1272) = 24$.

$$4032 = 2^6 \times 3^2 \times 7^1$$

$$1272 = 2^3 \times 3^1 \times 53^1$$

$$\text{gcd} = 2^3 \times 3^1 = 24$$

Applications.

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

Greatest Common Divisor

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

Euclid's algorithm. [Euclid 300 BCE]

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

← base case

← reduction step,
converges to base case

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

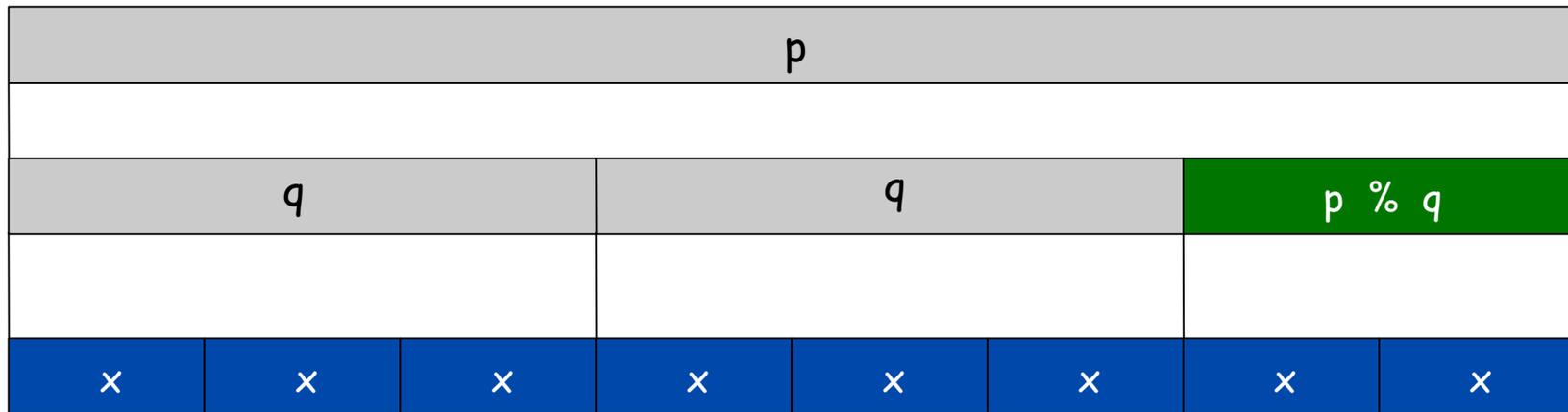
$$4032 = 3 \times 1272 + 216$$

Euclid's Algorithm

GCD. 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



$$\begin{aligned} p &= 8x \\ q &= 3x \end{aligned}$$

$$\text{gcd}(p, q) = \text{gcd}(3x, 2x) = x$$

Euclid's Algorithm

GCD. 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

Recursive program

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

← base case

← reduction step

Possible debugging challenges with recursion

Missing base case.

```
public static double BAD(int N)
{
    return BAD(N-1) + 1.0/N;
}
```

No convergence guarantee.

```
public static double BAD(int N)
{
    if (N == 1) return 1.0;
    return BAD(1 + N/2) + 1.0/N;
}
```

Both lead to INFINITE RECURSIVE LOOP (bad news).

Try it!

← so that you can recognize and deal with it if it later happens to you

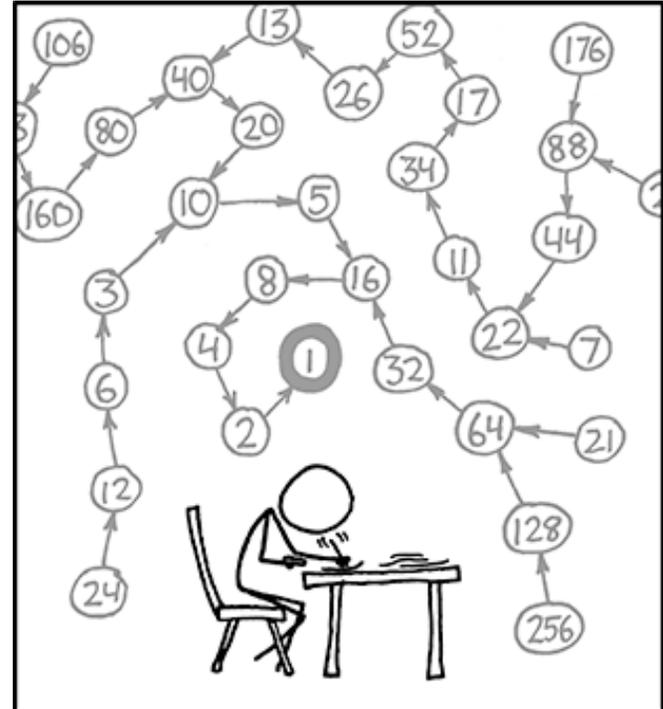
Collatz Sequence

Collatz sequence.

- If n is 1, stop.
- If n is even, divide by 2.
- If n is odd, multiply by 3 and add 1.

Ex. 35 106 53 160 80 40 20 10 5 16 8 4 2 1.

```
public static void collatz(int N)
{
    Stdout.print(N + " ");
    if (N == 1) return;
    if (N % 2 == 0) collatz(N / 2);
    collatz(3*N + 1);
}
```

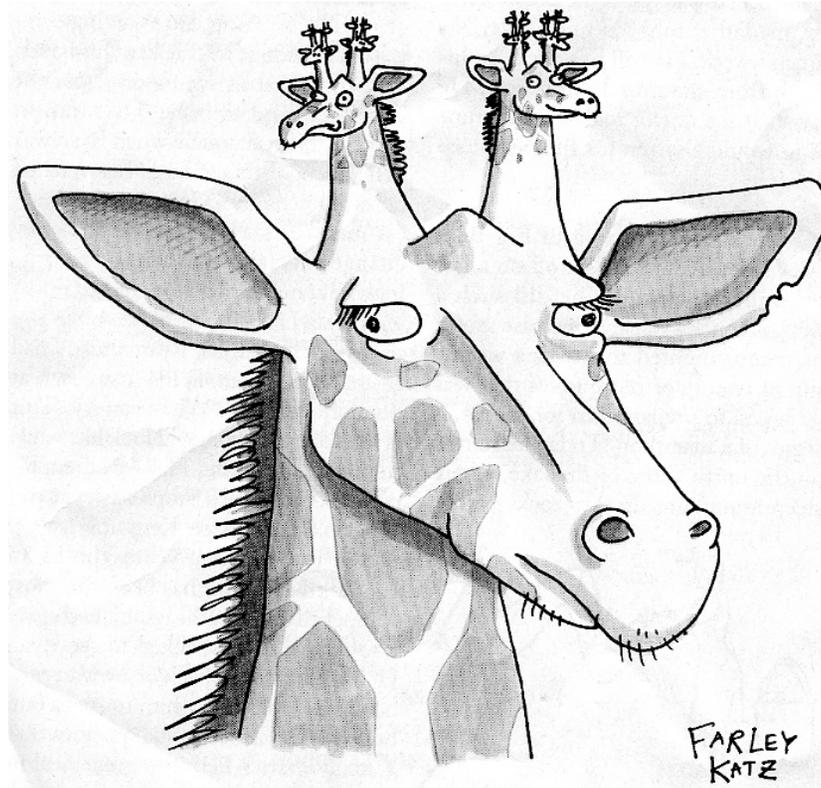


THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

No one knows whether or not this function terminates for all N (!)

[usually we decrease N for all recursive calls]

Recursive Graphics



New Yorker Magazine, August 11, 2008

The New York Times



Design Life Now at the Cooper-Hewitt National Design Museum includes this toy from the New York company Kidrobot.

Fruits of Design, Certified Organic

It's Triennial time at the Cooper-Hewitt National Design Museum. This means that the former Andrew Carnegie mansion is up to its neck in mostly American design from the last three years. Like its predecessors, "Design Life Now," the museum's third National Design Triennial, is a crowd-pleaser that illuminates a volatile, contradictory, ever-expanding field but fails to do it evenly.

ROBERTA SMITH
ART REVIEW
The exhibition has been organized by the Cooper-Hewitt curators Barbara Shorrock, Ellen Lapson and Matilda McQuaid and a guest, Brooke Hodge, a curator at the Museum of Contemporary Art, Los Angeles.

Once again the Triennial answers the question "What's design?" with the creative catchall "What's not?" Covering so many bases is equivalently, it never gets around to tackling the weightier question of "What is good design?" or, more to the point, "What is design good for?" It refuses to take sides on the issue of whether design should aim for social or environmental benefits or serve a relatively decorative purpose. Still, the show's benefits are many, even if you have to work for them.

The displays here range from genius to witless, delightful to disappointing. They cover life-extending innovations, completely functional reinventions of recovered items (far too many of which trace to Surrealism) and more varieties of recycling that you can easily count. Fashion, building materials, furniture, toys, theatrical sets, jewelry and textiles, medical and military hardware, all qualify as design according to this exhibition. The main point comes across loud and clear: design permeates every aspect of contemporary life. Everything that exists is designed, whether natural or cultural. And while all of nature's designs are intelligent, whether you go by Darwin or the Bible, the human kind are much

Continued on Page 51



Left: Show for The New York Times From "The Yale Book of Quotations" to "Postcards From Mars," a selection of the best holiday books.

The Gifts to Open Again and Again

I've made my list, and I'm checking it twice. It's a list of the qualities that make the ideal holiday book, and after carefully considering the books of Christmas past, I have come up with some guidelines. A gift book should either be so surprising or a big surprise: the one you always wanted or the one you never knew you wanted. It should be either expensive and large or cheap and small. It should be high-minded or totally frivolous. And no matter what, it should not require sustained attention, which is impossible during the yuletide season. My gift selections, chosen entirely at random but with exquisite taste, satisfy at least two of these requirements.

WILLIAM GRAMES
BOOKS
Let's open the big presents first. The season's whopper, in every way, is "New York 2001," the fifth installment in Robert A. M. Stern's architectural history of New York. The series starts in 1880, when it's most qualified as a skyscraper, and has now caught up to the new millennium. Taken together, the volumes make an enormous, endlessly fascinating family scrapbook for New Yorkers, who can recognize familiar pictures of the Flatiron Building and lead forward, through many hundreds of pages and thousands of photographs, to the growth of New York of the Lipstick Building, countless Trump projects and the new World Trade Center. At 1,500 pages and 18 pounds 12 ounces, "New York

Continued on Page 46

Divine and Devotee Meet Across Hinges

WASHINGTON—For toothache, dial St. Apollonia A.S.P. She's a bring relief in a flash. Keep St. Matthew, ex-banker, in mind in April, he'll help get your taxes in shape. Everyone knows that a prayer to St. Roch, protector from plague, is as good as a flu shot, and that lightning will never strike when St. Barbara's on the job.

HOLLAND
ART REVIEW
Most important, for dire and intractable problems—moral confusion, inextinguishable grief, sickness of soul—there's the Virgin. Day and night she's on the toll-free hot line of offering gentle attention and prudent advice.

To European Christians half a millennium ago, the Virgin and a raft of familiar saints were the exiled performed in a kind of celestial welfare system, available to all believers. And one quick way to access its benefits was through devotional paintings of the kind found in "Prayers and Portraits": Unfolding the Netherlands "diptychs" at the National Gallery of Art.

Probably outstanding in Western art comes closer to formal perfection than these pictures, produced by the Flemish Jan van Eyck, Rogier van der Weyden and Hugo van der Goes across an area that now encompasses the Netherlands, Belgium, Luxembourg and parts of



Prayers and Portraits: Including the Netherlands diptych. Two panels of an early 15th-century diptych by Maelfroid Sittow, left, are exhibited in an exhibition at the National Gallery of Art in Washington through Feb. 4.

Black, White and Read All Over Over

By RANDY KENNEDY
In one of Jorge Luis Borges's best-known short stories, "Pierre Menard, Author of the Quixote," a 20th-century French writer sets out to compose a verbatim copy of Cervantes's 17th-century masterpiece simply because he thinks he can, originally perhaps not being all that cracked up to do.

He manages two chapters worth of word, a spontaneous duplicate that Borges's narrator finds to be "infinitely richer" than the original because it contains all manner of new meanings and inflections, wrenched as it is from its proper time and context.

When a young Turkish artist named Serkan Ozdekan set out recently to practice his skills as a copyist—a scribe, as he says—his goals were a little less ambitious than channeling Cervantes. He simply wanted to draw and see printed, a faithful copy of all the type and pictures planned for a broadsheet page of this newspaper: this very page you are reading right now, which shows his version of his version of the page you are reading right now, which...

Mr. Ozdekan, a 30-year-old artist who lives and works in Istanbul, did not propose this exercise in handmade surrealism because of a particular love of calligraphy or newspapers or, for that matter, even drawing, which he admits he is not very good at. The

space-time-newsprint continuum.

Do not be alarmed: There has been no break in the



Black, White and Read All Over Over

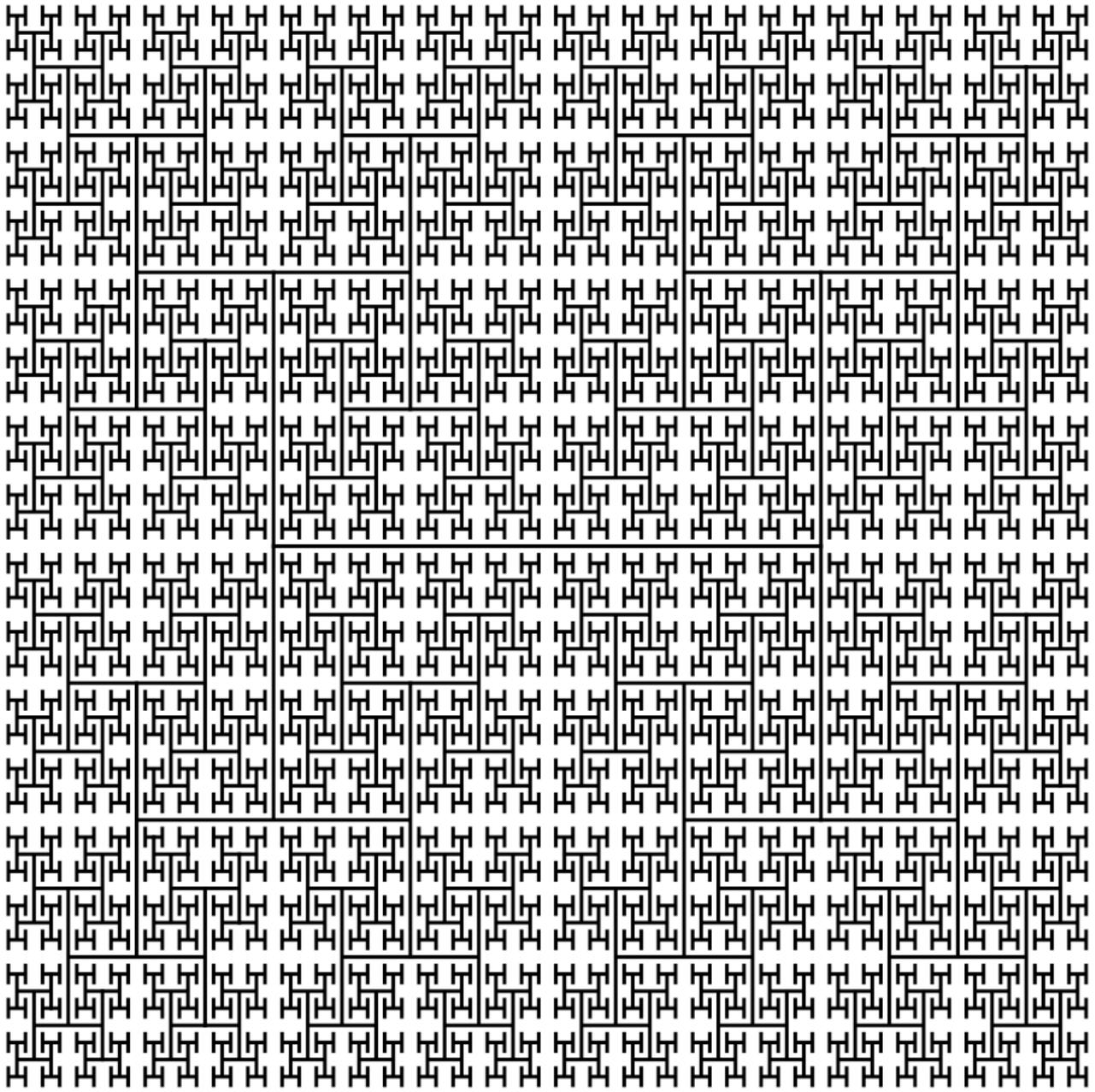
Do not be alarmed: There has been no break in the

Do not be alarmed: There has been no break in the

Do not be alarmed: There has been no break in the

Do not be alarmed: There has been no break in the

Do not be alarmed: There has been no break in the

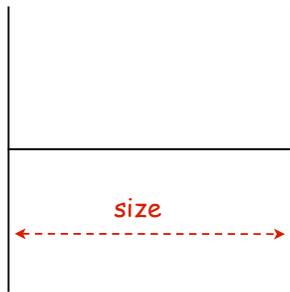


Htree

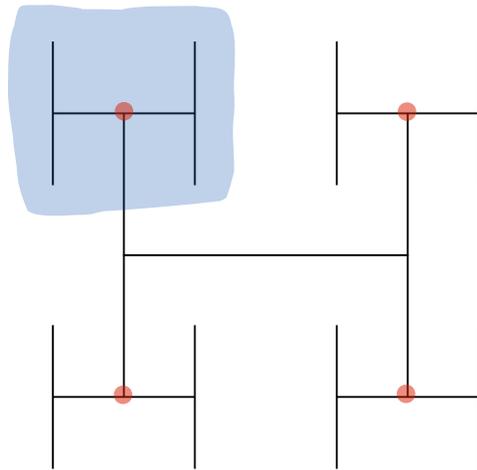
H-tree of order n .

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

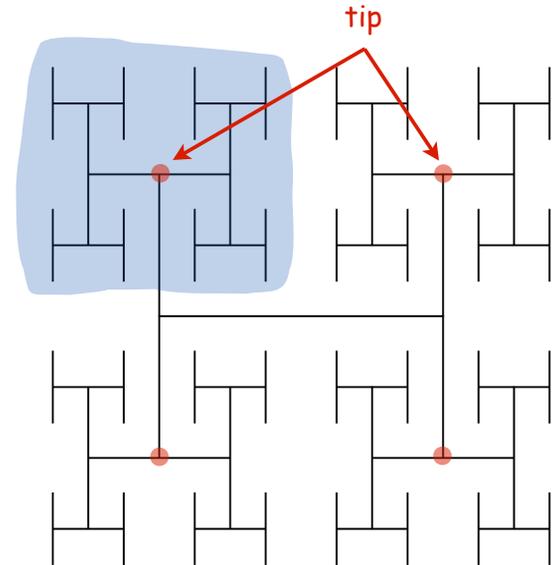
and half the size



order 1



order 2



order 3

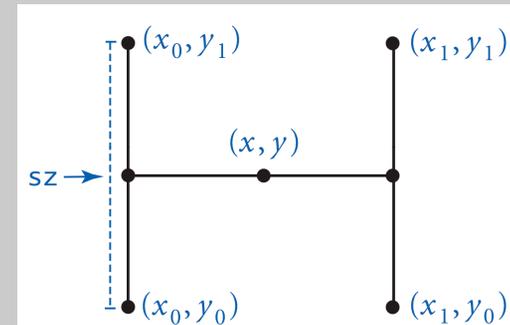
Htree in Java

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

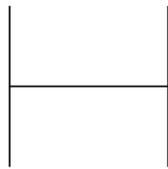
        draw(n-1, sz/2, x0, y0);
        draw(n-1, sz/2, x0, y1); ← recursively draw 4 half-size Hs
        draw(n-1, sz/2, x1, y0);
        draw(n-1, sz/2, x1, y1);
    }

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

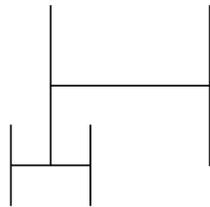


Animated H-tree

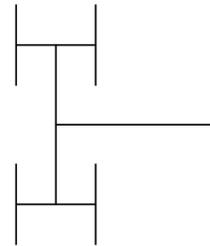
Animated H-tree. Pause for 1 second after drawing each H.



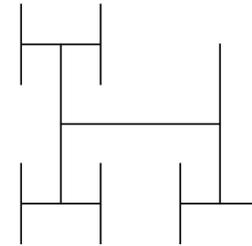
20%



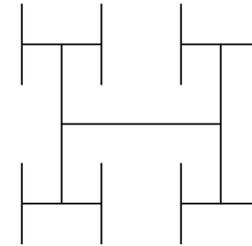
40%



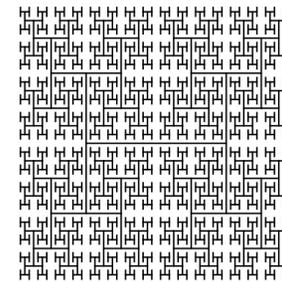
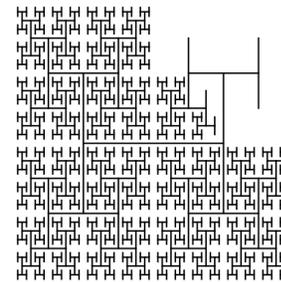
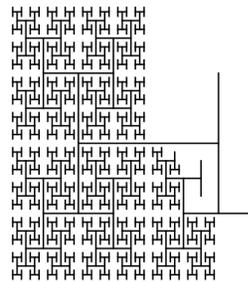
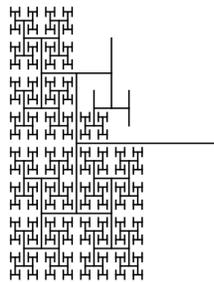
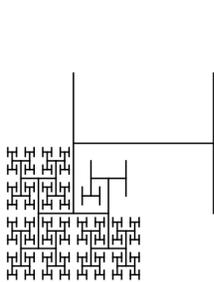
60%



80%



100%



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.

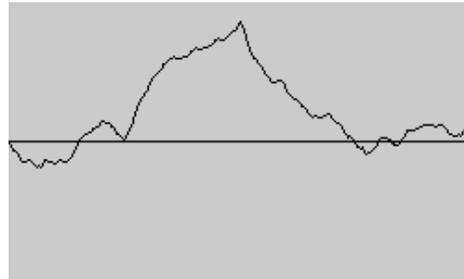
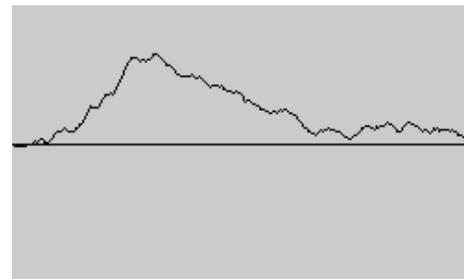
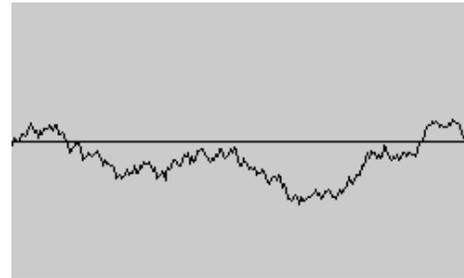
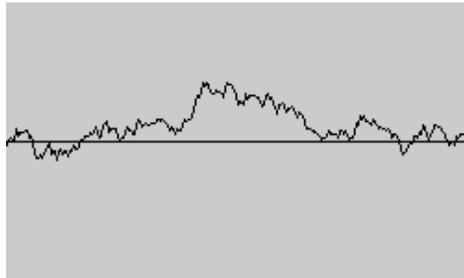
- FFT for signal processing.
- Parsers for programming languages.
- Multigrid methods for solving PDEs.
- **Quicksort and mergesort for sorting.**
- Hilbert curve for domain decomposition.
- Quad-tree for efficient N-body simulation.
- **Midpoint displacement method for fractional Brownian motion.**

Application: Fractional Brownian Motion

Fractional Brownian Motion

Physical process which models many natural and artificial phenomenon.

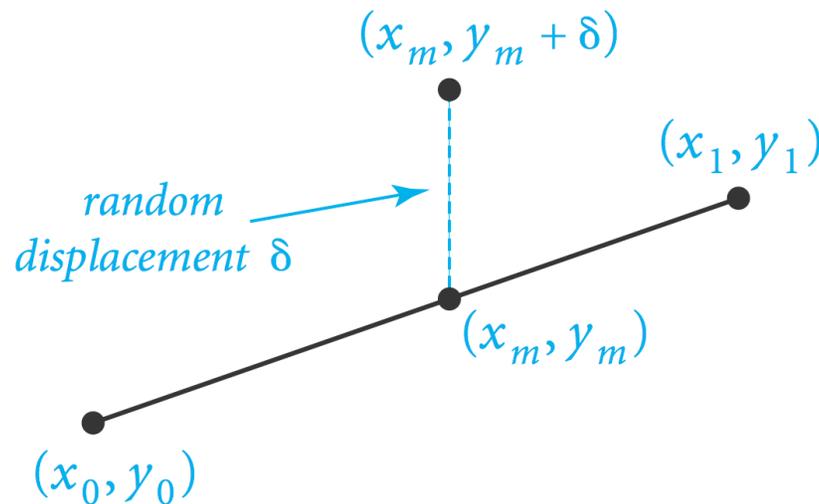
- Price of stocks.
- Dispersion of ink flowing in water.
- Rugged shapes of mountains and clouds.
- Fractal landscapes and textures for computer graphics.



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 $x_m = (x_0 + x_1)/2$ and $y_m = (y_0 + y_1)/2 + \delta$.
- Recur on the left and right intervals.



Simulating Brownian Motion: Java Implementation

Midpoint displacement method.

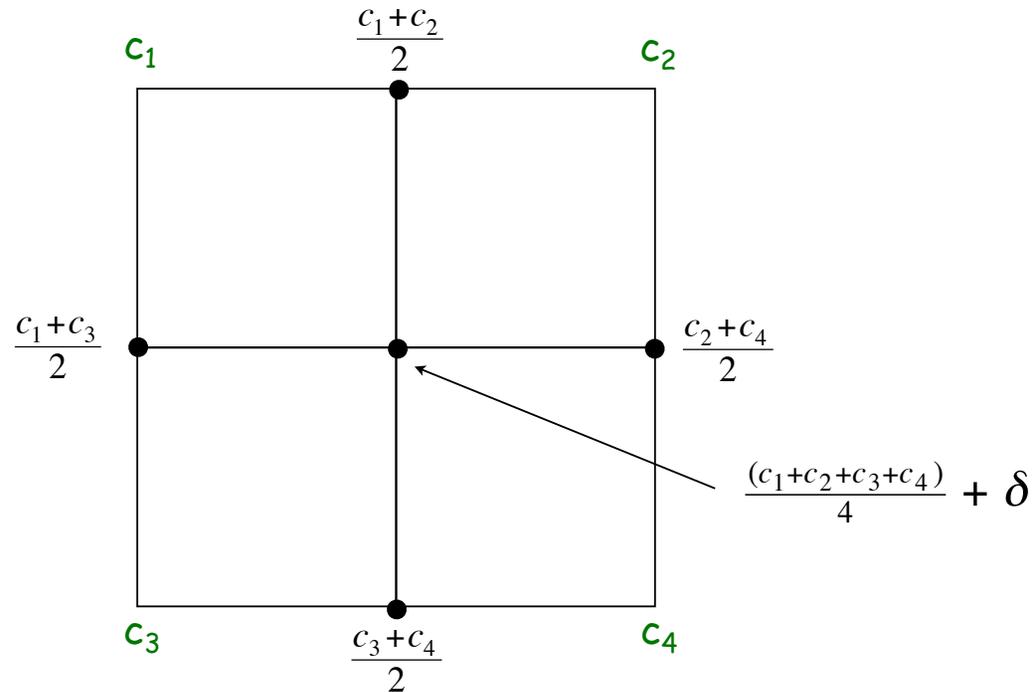
- Maintain an interval with endpoints (x_0, y_0) and (x_1, y_1) .
- Choose δ at random from Gaussian distribution.
- Divide the interval in half: Set $x_m = (x_0 + x_1)/2$ and $y_m = (y_0 + y_1)/2 + \delta$.
- Recur on the left and right intervals.

```
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
}
```

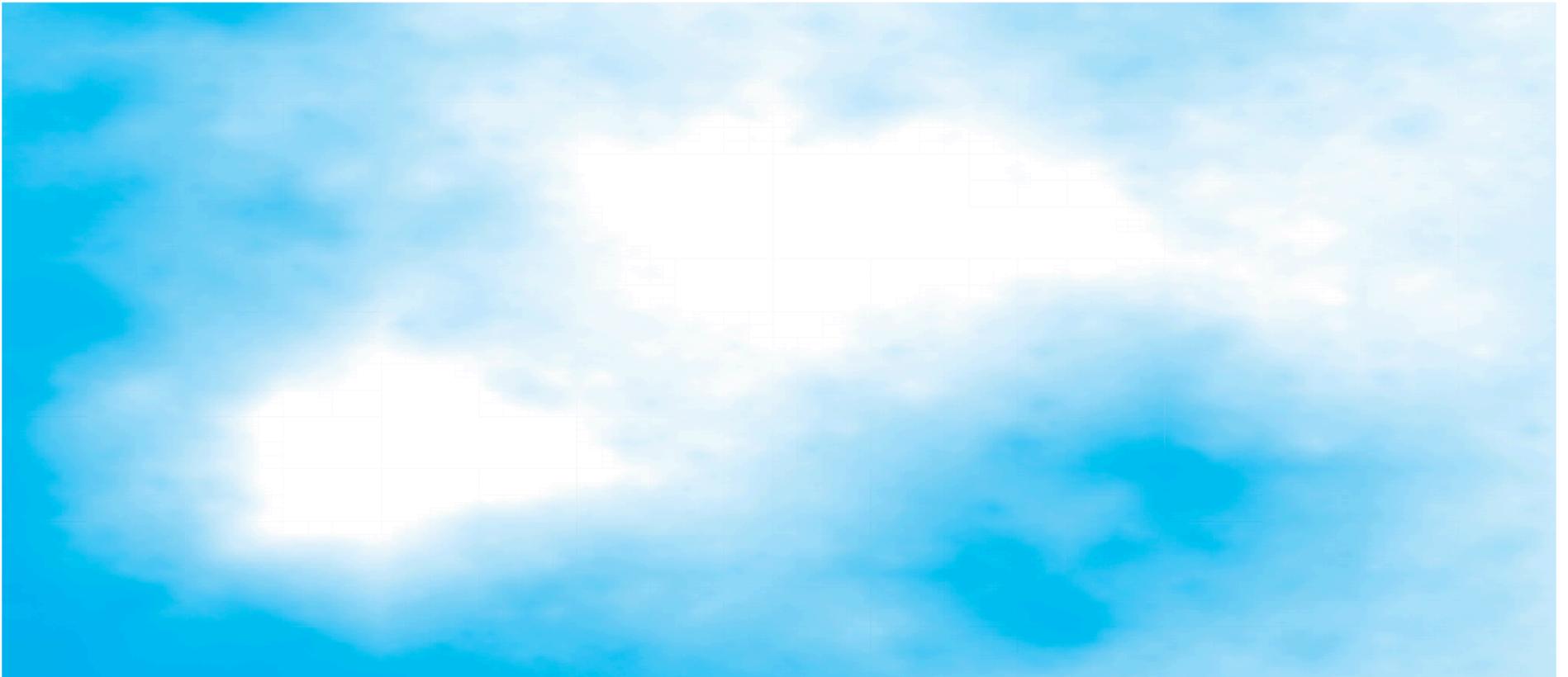
Plasma Cloud

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

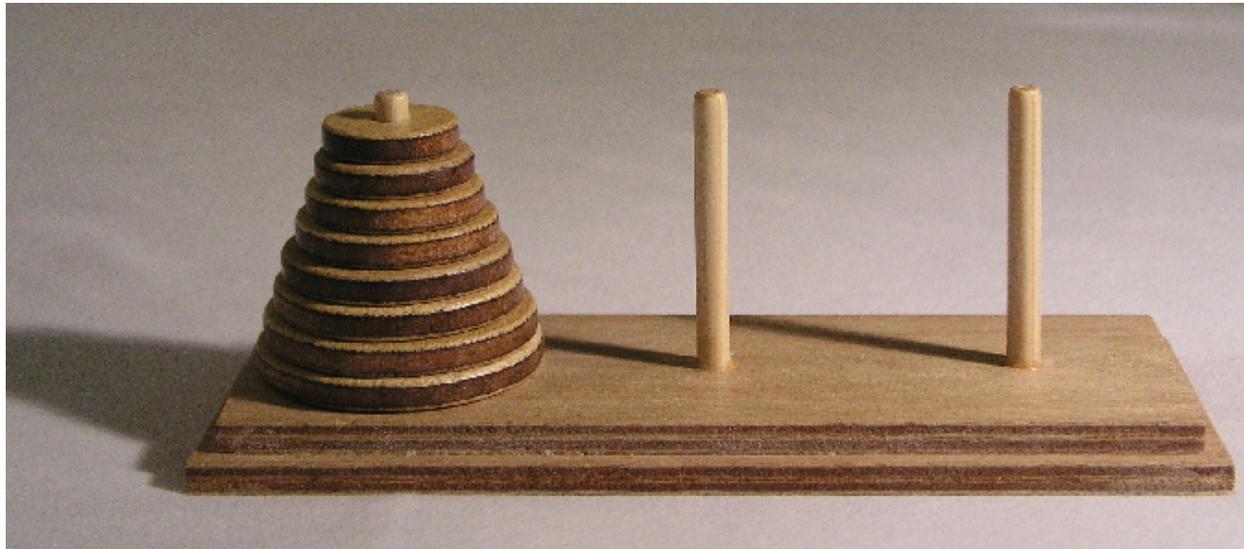


Brownian Landscape



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

Towers of Hanoi

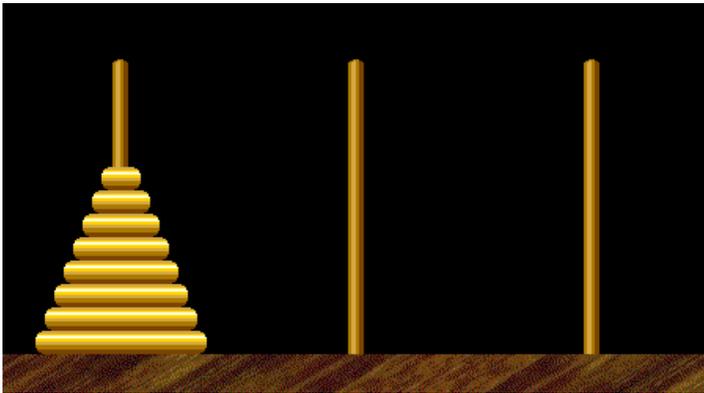


<http://en.wikipedia.org/wiki/Image:Hanoikleim.jpg>

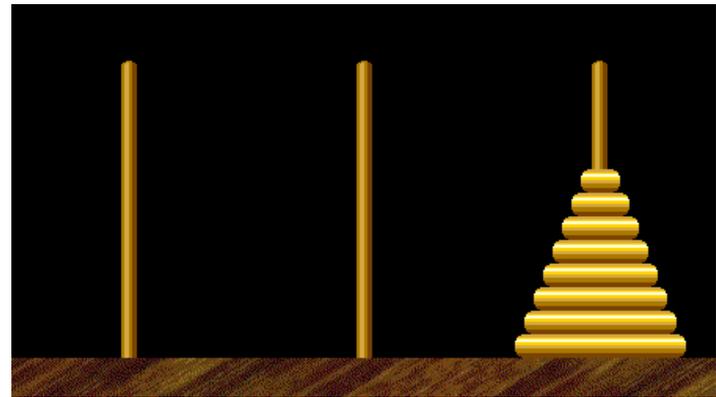
Towers of Hanoi

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

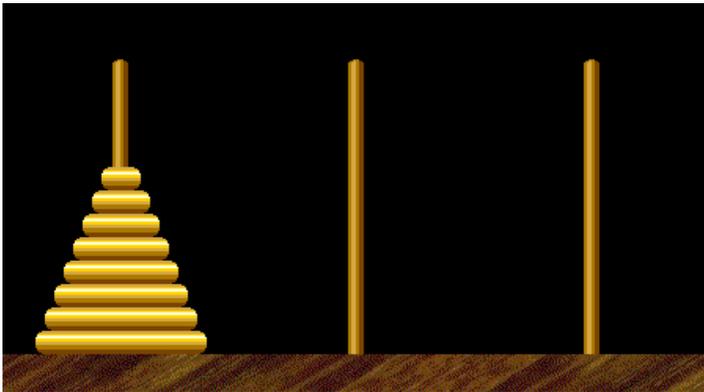


finish

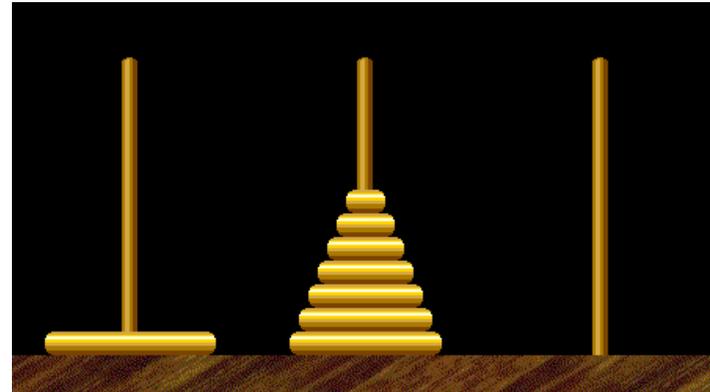


Edouard Lucas (1883)

Towers of Hanoi: Recursive Solution

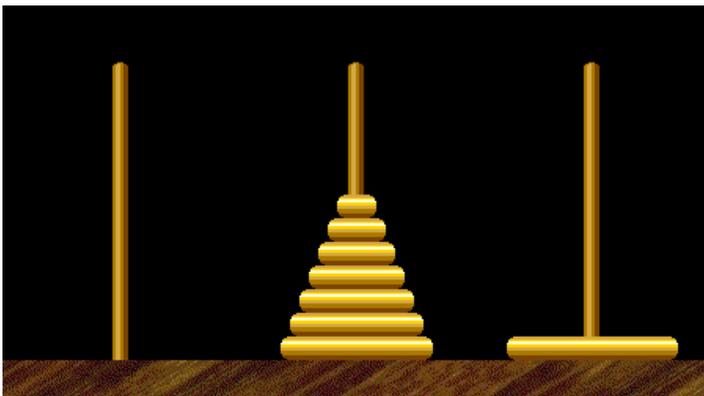


Move $n-1$ smallest discs right.

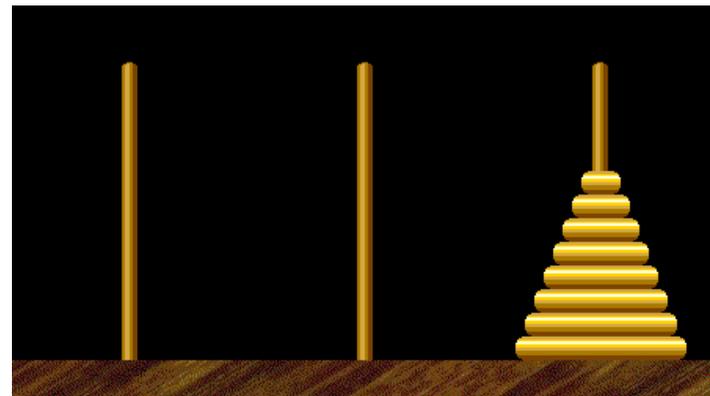


Move largest disc left.

← cyclic wrap-around



Move $n-1$ smallest discs right.



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

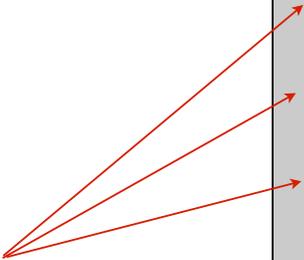
 smallest disc

Towers of Hanoi: Recursive Solution

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

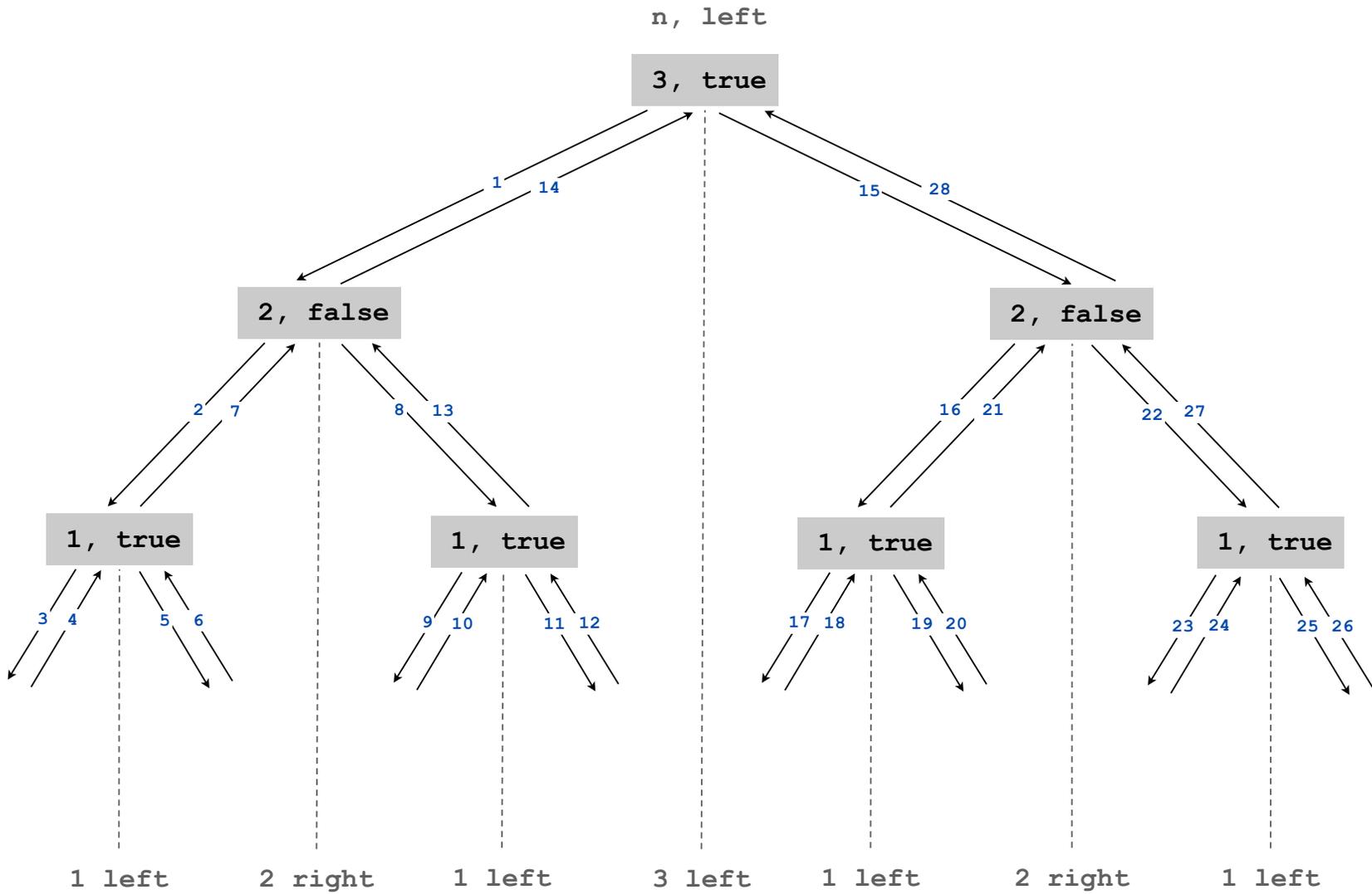
```
% java TowersOfHanoi 4
1 right
2 left
1 right
3 right
1 right
2 left
1 right
4 left
1 right
2 left
1 right
3 right
1 right
2 left
1 right
```

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.

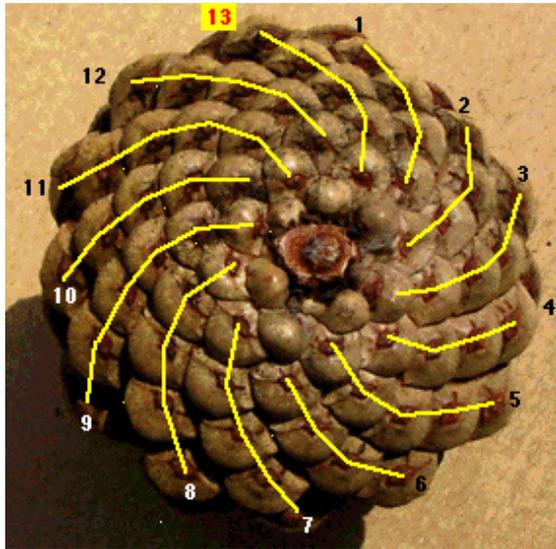
Recursive algorithm yields non-recursive solution!

- 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

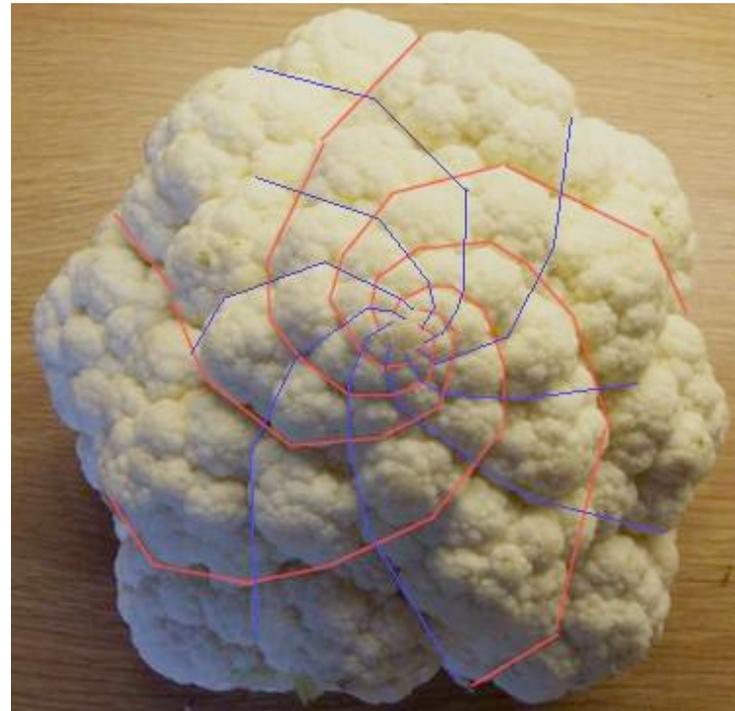
Recursive algorithm may reveal fate of world.

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

Fibonacci Numbers



pinecone

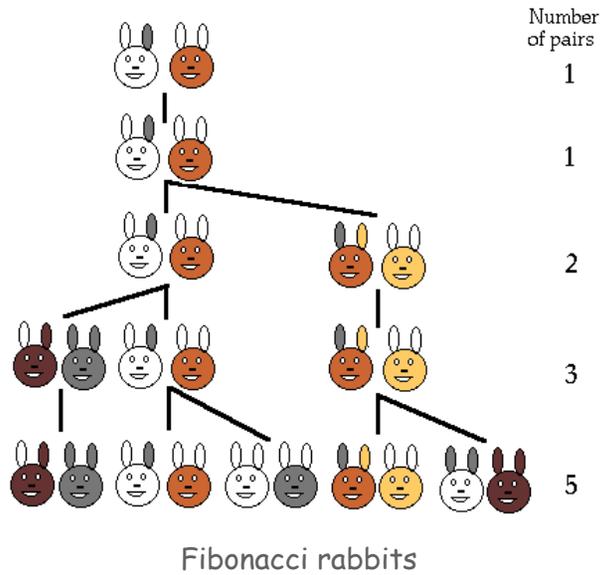


cauliflower

Fibonacci Numbers

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

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



L. P. Fibonacci
(1170 - 1250)

A Possible Pitfall With Recursion

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

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

FYI (classical math):

$$\begin{aligned} F(n) &= \frac{\phi^n - (1-\phi)^n}{\sqrt{5}} \\ &= \left\lfloor \phi^n / \sqrt{5} \right\rfloor \end{aligned}$$

ϕ = golden ratio ≈ 1.618

Ex: $F(50) \approx 1.2 \times 10^{10}$

A natural for recursion?

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

TEQ on Recursion 1.1 (difficult but important)

Is this an efficient way to compute $F(50)$?

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

TEQ on Recursion 1.2 (easy and also important)

Is this an efficient way to compute $F(50)$?

```
long[] F = new long[51];  
F[0] = 0; F[1] = 1;  
if (n == 1) return 1;  
for (int i = 2; i <= 50; i++)  
    F[i] = F[i-1] + F[i-2];
```

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.

Towers of Hanoi by W. A. Schloss.

Divide-and-conquer. Elegant solution to many important problems.

Exponential time.

- Easy to specify recursive program that takes exponential time.
- Don't do it unless you plan to (and are working on a small problem).