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.
- Linked data structures.
- A folder contains files and other folders.

Closely related to mathematical induction.

Greatest Common Divisor

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

Ex. gcd(4032, 1272) = 24.

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

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

$gcd = 2^3 \times 3^1 = 24$

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]

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

\[
gcd(4032, 1272) = \text{gcd}(1272, 216) = \text{gcd}(216, 192) = \text{gcd}(192, 24) = \text{gcd}(24, 0) = 24.
\]

4032 = 3 \times 1272 + 216

Reproductive Parts
M. C. Escher, 1948
Greatest Common Divisor

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

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

---

Java implementation.

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

---

Recursive Graphics

New Yorker Magazine, August 11, 2008
H-tree of order $n$.

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

**H-tree in Java**

```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);
        StdDraw.line(x1, y0, x1, y1);
        draw(n-1, sz/2, x0, y0);
        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.
**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.

**Towers of Hanoi: Recursive Solution**

1. Move n-1 smallest discs right.
2. Move largest disc left.
3. 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?
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);
    }
}

% 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 right
1 right
2 left
1 right
3 right
1 right
2 left
1 left

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

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.

Many important problems succumb to divide-and-conquer.
- FFT for signal processing.
- Parsers for programming languages.
- Multigrid methods for solving PDEs.
- Quicksort and mergersort for sorting.
- Hilbert curve for domain decomposition.
- Quad-tree for efficient N-body simulation.
- Midpoint displacement method for fractional Brownian motion.

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 Numbers and Nature

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 Possible Pitfall With Recursion

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

\[
F(n) = \begin{cases} 
0 & \text{if } n = 0 \\
1 & \text{if } n = 1 \\
F(n-1) + F(n-2) & \text{otherwise}
\end{cases}
\]

A natural for recursion?

Recursion Challenge 1 (difficult but important)

Q. Is this an efficient way to compute \(F(50)\)?

\[
\text{public static long } F(\text{int } n) \{ \\
\quad \text{if } (n == 0) \text{ return } 0; \\
\quad \text{if } (n == 1) \text{ return } 1; \\
\quad \text{return } F(n-1) + F(n-2); \\
\}
\]

A. No, no, no! This code is **spectacularly inefficient**.

Recursion Challenge 2 (easy and also important)

Q. Is this an efficient way to compute \(F(50)\)?

\[
\text{public static long(\text{int } n) \{ \\
\quad \text{long[ ] } F = \text{new long[n+1];} \\
\quad F[0] = 0; F[1] = 1; \\
\quad \text{for (int } i = 2; \ i <= n; \ i++) \\
\quad \quad F[i] = F[i-1] + F[i-2]; \\
\quad \text{return } F[n]; \\
\}}
\]

A. Yes. This code does it with 50 additions.

**Lesson.** Don’t use recursion to engage in exponential waste.

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.

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