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.

Greatest Common Divisor

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

Ex. gcd(4032, 1272) = 24.

\[
\begin{align*}
4032 &= 2^6 \cdot 3^1 \cdot 7^1 \\
1272 &= 2^3 \cdot 3^1 \cdot 53^1 \\
gcd &= 2^3 \cdot 3^1 = 24
\end{align*}
\]

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

Euclid’s algorithm. [Euclid 300 BCE]

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

Java implementation.

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

Greatest Common Divisor

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

```
<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p % q</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
```

\[
p = 8x \\
q = 3x \\
gcd(p, q) = x
\]
Recursive Graphics

Htree

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

Htree 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, y0, x0, y1);
        StdDraw.line(x1, y0, x1, y1);
        StdDraw.line(x0, y, x1, y);
        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

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 an empty peg or on top of a larger disc.

Towers of Hanoi demo

Towers of Hanoi Legend

Q. Is the 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

```java
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);
    }
}
```

Move n-1 smallest discs right.
Move largest disc left.
Move n-1 smallest discs right.

Towers of Hanoi: Recursive Solution

Subdivisions of ruler
every other move is smallest disc
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!

Fibonacci Numbers

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

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

A Possible Pitfall With Recursion

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

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

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

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

Recursion Challenge 1 (difficult but important)
Recursion Challenge 2 (easy and also important)

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

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

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

Lesson. Don't use recursion to engage in exponential waste.

Context. This is a special case of an important programming technique known as dynamic programming (stay tuned).

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.

Extra Slides

Collatz Sequence

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

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

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 \(\delta\) at random from Gaussian distribution.
- Set \(x_m = \frac{x_0 + x_1}{2}\) and \(y_m = \frac{y_0 + y_1}{2} + \delta\).
- 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
}
```

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.

Brownian Landscape


Robert Brown (1773-1858)