2.3 Recursion

Greatest Common Divisor

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

Example:
- Suppose \( p = 32 \) and \( q = 24 \)
- Integers that evenly divide both \( p \) and \( q \): 1, 2, 4, 8
  - So \( d = 8 \) (the largest)

How would you compute \( \gcd \)?

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.

Greatest Common Divisor

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

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

Base case: \( \text{gcd} \) converges to base case.

\[
\begin{align*}
\text{gcd}(4032, 1272) & = \text{gcd}(1272, 216) \\
& = \text{gcd}(216, 192) \\
& = \text{gcd}(192, 24) \\
& = \text{gcd}(24, 0) \\
& = 24.
\end{align*}
\]

\[
\begin{align*}
4032 & = 2^6 \times 3^2 \times 7^1 \\
1272 & = 2^3 \times 3^1 \times 5^2 \\
\text{gcd} & = 2^3 \times 3^1 = 24
\end{align*}
\]

Applications:
- Simplify fractions: \( \frac{1272}{4032} = \frac{53}{168} \).
- RSA cryptosystem: stay tuned
- History of algorithms.

Euclid, 300 BCE
Greatest Common Divisor

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

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

- base case
- reduction step, converges to base case

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \mod q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
</tbody>
</table>

$p = 8x$  
$q = 3x$  
$\text{gcd}(p, q) = x$

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 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);
    }
}
```
**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: Recursive Solution

- Move N-1 smallest discs to pole B.
- Move largest disc to pole C.
- Move N-1 smallest discs to pole C.

**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.

**Will computer algorithms help?**
Towers of Hanoi: Recursive Solution

```java
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");
    }
}
```

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!
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.
- 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.

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

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 \(x_{\text{mid}} = (x_0 + x_1)/2\) and \(y_{\text{mid}} = (y_0 + y_1)/2 + \Delta\).
- 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 \(\Delta\) 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.

```java
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);
}
```
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.

Fibonacci Numbers

Infinite series: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$
- A natural for recursion.

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

Possible Pitfalls With Recursion

Is recursion fast?
- Yes. We produced remarkably efficient program for gcd.
- No. Can easily write remarkably inefficient programs.

Fibonacci numbers:
$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

Observation: it takes a really long time to compute $F(40)$.

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

Spectacularly inefficient Fibonacci

Fibonacci Rabbits:

L. P. Fibonacci (1170 - 1250)

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.

\[
\begin{align*}
F(39) & \rightarrow F(38) \\
F(38) & \rightarrow F(37) \\
F(37) & \rightarrow F(36) \quad \text{and} \quad F(36) \quad \text{and} \quad F(36) \\
F(36) & \rightarrow F(35) \quad \text{and} \quad F(35) \quad \text{and} \quad F(35) \quad \text{and} \quad F(35)
\end{align*}
\]

F(0) is computed $165,580,141$ times.
331,160,281 function calls for $F(40)$.

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

Spectacularly inefficient Fibonacci
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 problems have elegant divide-and-conquer solutions.
- Adaptive quadrature.
- Quicksort.
- Integer arithmetic for RSA cryptography.
- Polynomial multiplication for signal processing.
- Quad-tree for efficient N-body simulation.