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 $d$ that evenly divides into $p$ and $q$.

**Euclid’s algorithm.** [Euclid 300 BCE]

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

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

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

\( \text{Gcd.} \) Find largest integer \( d \) that evenly divides into \( p \) and \( q \).

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

- base case
- reduction step, converged to base case

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p % q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 8x )</td>
<td>( 3x )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

Java implementation.

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

- base case
- reduction step, converged to base case

Recursive Graphics
Htree

H-tree of order n
- Draw an H of size and half the size
- 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 sx, double s, double y) {
        if (n == 0) return;
        double x0 = x - sx/2, x1 = x + sx/2;
        double y0 = y - s/2, y1 = y + s/2;
        StdDraw.line(x0, y, x1, y);
        StdDraw.line(x0, y0, x0, y1);
        StdDraw.line(x1, y0, x1, y1);
        draw(n-1, sx/2, x0, y0);
        draw(n-1, sx/2, x1, y1);
        draw(n-1, sx/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 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

- Move n-1 smallest discs right.
- Move largest disc left.
- Move n-1 smallest discs right.
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); // move (n-1) discs
        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

Towers of Hanoi: Recursion Tree

- 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 mergesort 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, 2, 3, 5, 8, 13, 21, 34, ...

A natural for recursion?

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

Recursion Challenge 1 (difficult but important)

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

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

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

A. F(50) is called once.
F(49) is called once.
F(48) is called 2 times.
F(47) is called 3 times.
F(46) is called 5 times.
F(45) is called 8 times.
...
F(1) is called 12,586,269,025 times.

Recursion tree for naïve Fibonacci function

F(50)

F(49)

F(48)

F(47)

F(46)

F(45)

F(44)

...

Recursion Challenge 2 (easy and also important)

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

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

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

A. F(50) is called once.
F(49) is called once.
F(48) is called 2 times.
F(47) is called 3 times.
F(46) is called 5 times.
F(45) is called 8 times.
...
F(1) is called 12,586,269,025 times.

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.

**FYI: classic math**

```
f(n) = \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}}
\phi = \frac{1 + \sqrt{5}}{2} \approx 1.618
```

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

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.

```java
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);
}
```

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.

```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);  // variance halves at each level
}
```

Simulating Brownian Motion: Java Implementation

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.

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.

\[ c_1 + c_2 + c_3 + c_4 = 4 \]

\[ c_2 + c_3 = \frac{s^2}{2} \]

\[ c_1 + c_2 = \frac{s^2}{2} + \delta \]