Recursion
Recursive Definitions

Recursive Definition

A definition is called *recursive* if the object is defined in terms of itself.

Base Case

Recursive definitions require a *base case* at which to either initiate or terminate the definition.

Otherwise recursive definitions would be circular.
Recursive Programs

A program is called *recursive* if the program calls itself.

Base Case

Recursive programs require a *base case* at which to either initiate or terminate the program.

Otherwise recursive programs would never terminate.
Fractals
Fractals

Recursion Made Visible

Recursive Turtle Programs

- Sierpinski Triangle
- Koch Snowflake
- Fractal Trees
Turtles

What the Turtle Knows

- CURRENT_POSITION = (x, y) location -- where she is
- CURRENT_HEADING = (u,v) vector -- where she is going

(Slightly more than the average professor.)

Turtle Commands (Rice LOGO)

- FORWARD(D) -- change position, and draw a straight line of length D
- MOVE(D) -- same as FORWARD without drawing a line
- TURN(A) -- change heading by A, but do not change position
- RESIZE(S) -- change step size by a factor of S, but do not change position or direction
- Usual collection of control commands -- loops, conditionals,…

Turtle Programs

- Finite sequence of FORWARD, TURN, AND RESIZE commands
- Draws a piecewise linear curve
More Examples of Recursion

1. Sums and Differences
2. IQ-Tests
3. Binomial Coefficients
   -- Pascal’s Triangle
4. Fibonacci Numbers
   -- Cat Problem
5. Tower of Hanoi
   -- Animation
6. Natural Numbers
7. Polynomials
8. Rooted Binary Trees
   -- Single Vertex
   -- Root connected to two roots
IQ Tests

- 6, 6, 6, 6, 6, 6, ?
- 4, 7, 10, 13, 16, 19, ?
- 3, 13, 29, 51, 79, 113, ?
- 2, 3, 12, 35, 78, 147, ?
Forward Differencing

Forward Differencing

• $\Delta F(k) = F(k+1) - F(k)$

• $\Delta^{n+1}F(k) = \Delta\left(\Delta^n F\right)(k)$

Properties of First Difference

• $\Delta(F + G)(k) = \Delta F(k) + \Delta G(k)$

• $\Delta(cF)(k) = c(\Delta F(k))$
Forward Differencing Monomials

Theorem 1

\((\Delta x^p)(k) = (k + 1)^p - k^p = pk^{p-1} + \text{lower order terms}\)

Proof: Binomial Theorem.

Theorem 2

\((\Delta^n x^p)(k) = p \cdots (p - n + 1)k^{p-n} + \text{lower order terms} \quad n \leq p\)

Proof: Induction on \(n\).

\((\Delta^{n+1} x^p)(k) = \Delta \left(\Delta^n x^p\right)(k)\)

\[= \Delta \left(p \cdots (p - n + 1)k^{p-n} + \text{lower order terms}\right)\]
Forward Differencing Polynomials

Corollaries

• $(\Delta^p x^p)(k) = p! \quad (constant)$

• $(\Delta^n x^p)(k) = 0 \quad n > p$

• $\Delta^p (a_p x^p + \cdots + a_1 x + a_0)(k) = p! a_p$
### IQ Tests -- Revisited

<table>
<thead>
<tr>
<th>$F$</th>
<th>3, 13, 29, 51, 79, 113, ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta F$</td>
<td>10, 16, 22, 28, 34, ?</td>
</tr>
<tr>
<td>$\Delta^2 F$</td>
<td>6, 6, 6, 6, ?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$F$</th>
<th>2, 3, 12, 35, 78, 147, ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta F$</td>
<td>1, 9, 23, 43, 69, ?</td>
</tr>
<tr>
<td>$\Delta^2 F$</td>
<td>8, 14, 20, 26, ?</td>
</tr>
<tr>
<td>$\Delta^3 F$</td>
<td>6, 6, 6, ?</td>
</tr>
</tbody>
</table>
IQ Tests -- Revisited

\[ F = 3, 13, 29, 51, 79, 113, 153? \]
\[ \Delta F = 10, 16, 22, 28, 34, 40? \]
\[ \Delta^2 F = 6, 6, 6, 6, 6? \]

\[ F = 2, 3, 12, 35, 78, 147, 248? \]
\[ \Delta F = 1, 9, 23, 43, 69, 101? \]
\[ \Delta^2 F = 8, 14, 20, 26, 32? \]
\[ \Delta^3 F = 6, 6, 6, 6? \]
Exponential Sequences

\[ F \quad 1, \ 2, \ 4, \ 8, \ 16, \ 32, \ ? \]

\[ \Delta F \quad 1 \ 2 \ 4 \ 8 \ 16 \ ? \]
Exponential Sequences

\[ F \quad 1, \quad 2, \quad 4, \quad 8, \quad 16, \quad 32, \quad 64 \]

\[ \Delta F \quad 1 \quad 2 \quad 3 \quad 8 \quad 16 \quad 32 \]

\[ \Delta F = F_{n+1} - F_n = 2^{n+1} - 2^n = 2^n (2 - 1) = 2^n \]

Theorem

*There is no polynomial \( p(x) \) for which \( 2^n = p(n) \) for all \( n \).*
Differences and Derivatives

**Difference**

\[(\Delta x^p)(k) = pk^{p-1} + \text{lower order terms}\]

\[(\Delta^p x^p)(k) = p!\]

\[(\Delta 2^p)(k) = 2^k\]

**Derivative**

\[\frac{d}{dx} x^p = px^{p-1}\]

\[\frac{d^p}{d^p x} x^p = p!\]

\[\frac{d}{dx} e^x = e^x\]
**Pounce the Cat**

My cat, Pounce, can walk up steps either one or two at a time. There are 21 steps from the first to the second floor of Duncan Hall. In how many different ways can Pounce chase a mouse up the steps from the first to the second floor of Duncan Hall?
Fibonacci Sequences

Fibonacci Recurrence

• $f_1 = f_2 = 1$ (Base Cases)
• $f_{n+1} = f_n + f_{n-1}$ (Recursion)

Fibonacci Sequence

• $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...$
Fibonacci Sequences

Fibonacci Sequence

\begin{align*}
f &= 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots \\
\Delta f &= 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
\end{align*}

\[ \Delta f = f_{n+1} - f_n = (f_n + f_{n-1}) - f_n = f_{n-1} \]

Theorem

There is no polynomial \( p(x) \) for which \( f_n = p(n) \) for all \( n \).
Recursive Computation of Fibonacci Numbers

Number of Additions = Exponential
Iterative Computation of Fibonacci Numbers

Number of Additions = 2(n − 2)
Iterative Computation of Binomial Coefficients

$$\begin{align*}
\text{Number of Additions} &= \sum_{k=1}^{n+1} k = \frac{(n+1)(n+2)}{2}
\end{align*}$$
Neville’s Algorithm

Linear Interpolation  (Base Case)

\[ I_{D_2}(x) = \frac{x_2 - x}{x_2 - x_1} y_1 + \frac{x - x_1}{x_2 - x_1} y_2 \]

Recursion

\[ I_{D_n}(x) = \frac{x_n - x}{x_n - x_1} I_{D_n^-}(x) + \frac{x - x_1}{x_n - x_1} I_{D_n^+}(x) \]

- \( D_n = (x_1, y_1), \ldots, (x_n, y_n) \)
- \( D_n^- = (x_1, y_1), \ldots, (x_{n-1}, y_{n-1}) \)
- \( D_n^+ = (x_2, y_2), \ldots, (x_n, y_n) \)
Polynomial Interpolation and Neville’s Algorithm

Questions

i. How Fast is Neville’s Algorithm?

ii. What is the Best Way to Program Neville’s Algorithm?
Neville’s Algorithm -- Recursive Implementation

\[ I_{1,\ldots,n}(x) \]

\[ I_{1,\ldots,n-1}(x) \quad I_{2,\ldots,n}(x) \]

\[ I_{1,2}(x) \quad \cdots \quad I_{n-1, n}(x) \]

\[ y_1 \quad y_2 \quad \cdots \quad y_{n-1} \quad y_n \]

\( n-1 \) Levels \( \leftrightarrow \) \( 2^n - 2 \) multiplications
Neville’s Algorithm -- Iterative Implementation

\[ I_{1,2,3,4} \]

\[ I_{1,2,3} \quad I_{2,3,4} \]

\[ I_{1,2} \quad I_{2,3} \quad I_{3,4} \]

\[ y_1 \quad y_2 \quad y_3 \quad y_4 \]

\[ n - 1 \text{ Levels} \leftrightarrow 2 \sum_{k=1}^{n-1} k = (n - 1)n \text{ multiplications} \]
Tower of Hanoi

http://www.cut-the-knot.org/recurrence/hanoi.shtml

http://www.dynamicdrive.com/dynamicindex12/towerhanoi.htm

http://www.mathsisfun.com/games/towerofhanoi.html
Tower of Hanoi

Recursive Solution

- 1 Ring -- Trivial
- $N + 1$ Rings -- Solve $N$ Ring Problem Twice

Number of Moves

- $M(1) = 1$
- $M(N + 1) = 2M(N) + 1$
  -- $M(N) = 2^N - 1$
Program Correctness

Loop Invariant

• A *property* or *number* that is the same before and after each iteration of a loop.

• Used to prove program correctness.

• Proofs proceed by induction on the number of iterations.
Example

**Integer Division**

- Input: \( m, n = \text{integers} \)
- Output: \( q, r = \text{integers} \) (quotient and remainder)
  
  \[ n = mq + r \quad \text{with} \quad 0 \leq r < m \]

**Algorithm**

\[ q = 0 \]
\[ r = n \]

While \( r \geq m \) 

\[ q = q + 1 \quad \text{Loop Invariant} \]
\[ n = mq + r \]
\[ r = r - m \]
Proofs

Proof of Loop Invariance

By induction on the number of iteration of the loop.

Base Case: \( q = 0 \) and \( r = n \) \( \Rightarrow \) \( n = mq + r \).

Induction: Suppose that \( n = mq + r \) after \( k \) iterations of the loop.

Must show that \( n = mq + r \) after \( k + 1 \) iterations of the loop.

But after \( k + 1 \) iterations of the loop:

\[
(mq + r)_{k+1} = mq_{k+1} + r_{k+1} \\
= m(q_k + 1) + r_k - m \\
= mq_k + r_k \\
= n.
\]

(inductive hypothesis)

Proof of Program Correctness

1. The loop will terminate with \( r < m \).
2. When the loop terminates \( n = mq + r \). (Loop Invariance)
**Example**

*GCD -- Greatest Common Divisor*

- Input: $m, n =$ integers
- Output: $GCD(m,n)$

*Euclidean Algorithm*

\[ x = m \]
\[ y = n \]

While $y \neq 0$

\[ r = \text{remainder of } x \text{ divided by } y \]  
\[ \text{(use division algorithm)} \]

\[ x = y \]
\[ y = r \]

Output: $x$ is $GCD(m,n)$
Proof of Loop Invariance

By induction on the number of iteration of the loop.

Base Case: \( x = m \) and \( y = n \) \( \implies \) \( GCD(x, y) = GCD(m, n) \).

Induction: Suppose that \( GCD(x, y) = GCD(m, n) \) after \( k \) iterations of the loop.

Must show that \( GCD(x, y) = GCD(m, n) \) after \( k+1 \) iterations.

After the first line of the \((k+1)^{st}\) iteration of the loop:
\[
x = yq + r \implies r = x - yq \implies GCD(y, r) = GCD(x, y)
\]
and by the inductive hypothesis
\[
GCD(x, y) = GCD(m, n) \quad \text{so} \quad GCD(y, r) = GCD(m, n).
\]

But after the \((k+1)^{st}\) iteration of the loop:
\[
x = y \quad \text{and} \quad y = r
\]
so
\[
GCD(x, y) = GCD(y, r) = GCD(m, n).
\]
Loop

While $y \neq 0$

\[ r = \text{remainder of } x \text{ divided by } y \]
\[ (\text{use division algorithm}) \]

\[ x = y \]
\[ y = r \]

Loop Invariant

\[ GCD(x, y) = GCD(m, n) \]

Proof of Program Correctness

1. The loop will terminate when $y = r = 0$.

2. Therefore before the last iteration of the Loop, $y$ divides $x$. Hence just before the start of the final iteration
   a. $y = GCD(x, y)$
   b. $y = GCD(m, n)$ (Loop Invariance)

3. But after the final iteration: $x = y$
   so after the final iteration: $x = GCD(m, n)$.