<u>Recursion</u>

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

Recursive Programs

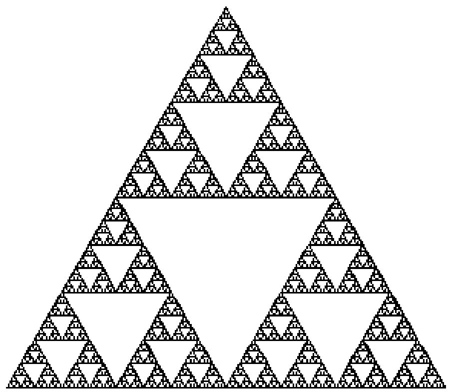
A program is called *recursive* if the program calls itself.

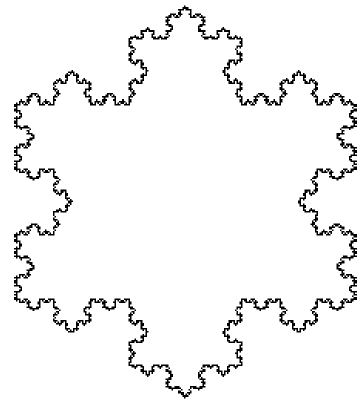
Base Case

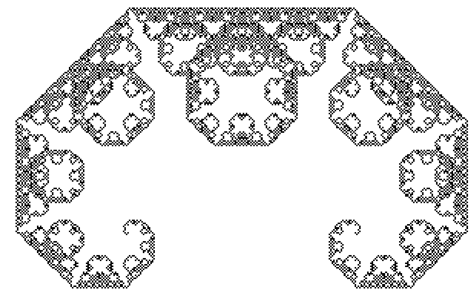
Recursive programs require a *base case* at which to either initiate or terminate the program.

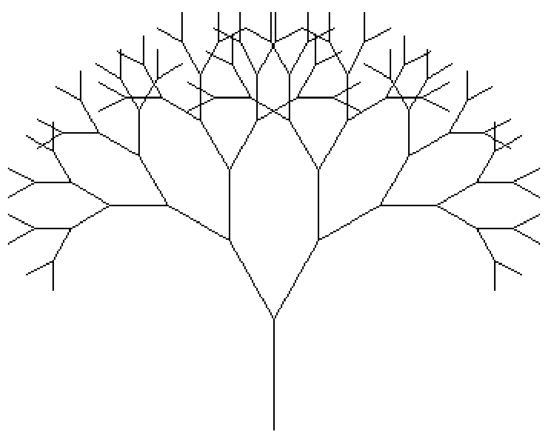
Otherwise recursive programs would never terminate.

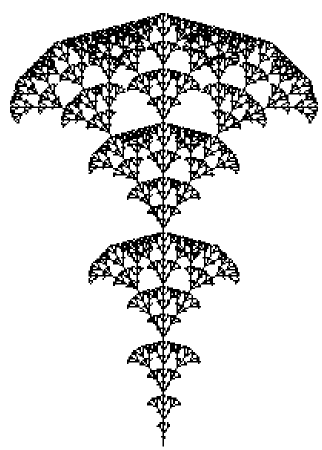
Fractals

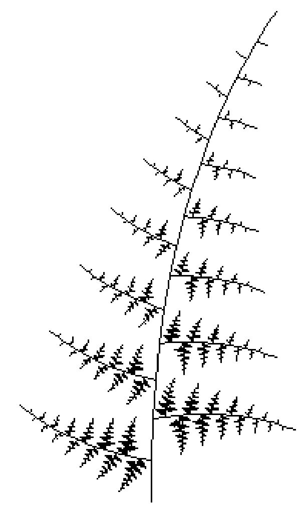












<u>Fractals</u>

Recursion Made Visible

Recursive Turtle Programs

- Sierpinski Triangle
- Koch Snowflake
- Fractal Trees

<u>Turtles</u>

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

3. Fibonacci Numbers
   -- Cat Problem

4. Tower of Hanoi
   -- Animation

5. Natural Numbers

6. Polynomials

7. 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, ?

<u>Forward Differencing</u>

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

- $\left(\Delta(F+G)\right)(k) = \Delta F(k) + \Delta G(k)$
- $\Delta(c\,F)(k) = c\left(\Delta F(k)\right)$

Forward Differencing Monomials

Theorem 1

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

Proof: Binomial Theorem.

Theorem 2

$$(\Delta^n x^p)(k) = p \cdots (p-n+1)k^{p-n} + lower\ order\ terms \qquad n \le p$$

Proof: Induction on n.

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

$$= \Delta\left(p \cdots (p-n+1)k^{p-n} + lower\ order\ terms\right)$$

<u>Forward Differencing Polynomials</u>

Corollaries

- $(\Delta^p x^p)(k) = p!$ *(constant)*

- $(\Delta^n x^p)(k) = 0$ $n > p$

- $\Delta^p (a_p x^p + \cdots + a_1 x + a_0)(k) = p!\, a_p$

IQ Tests -- Revisited

F	3,	13,	29,	51,	79,	113,	**?**
ΔF	10	16	22	28	34	**?**	
$\Delta^2 F$	6	6	6	6	**?**		

F	2,	3,	12,	35,	78,	147,	?
ΔF	1	9	23	43	69	**?**	
$\Delta^2 F$	8	14	20	26	**?**		
$\Delta^3 F$	6	6	6	**?**			

IQ Tests -- Revisited

F	3,	13,	29,	51,	79,	113,	**153?**
ΔF	10	16	22	28	34	**40?**	
$\Delta^2 F$	6	6	6	6	**6?**		

F	2,	3,	12,	35,	78,	147,	**248**?
ΔF	1	9	23	43	69	**101?**	
$\Delta^2 F$	8	14	20	26	**32?**		
$\Delta^3 F$	6	6	6	**6?**			

Exponential Sequences

$F \quad 1, \quad 2, \quad 4, \quad 8, \quad 16, \quad 32, \quad \mathbf{?}$

$\Delta F \quad 1 \quad 2 \quad 4 \quad 8 \quad 16 \quad \mathbf{?}$

Exponential Sequences

F	1,	2,	4,	8,	16,	32,	**64**
ΔF	1	2	3	8	16	**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	*Derivative*
$(\Delta x^p)(k) = p\,k^{p-1} + lower\ \ order\ \ terms$	$\frac{d}{dx}x^p = px^{p-1}$
$(\Delta^p x^p)(k) = p!$	$\frac{d^p}{d^p x}x^p = p!$
$(\Delta 2^p)(k) = 2^k$	$\frac{d}{dx}e^x = e^x$

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

- $1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89, \ldots$

Fibonacci Sequences

Fibonacci Sequence

$f \quad 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89, \ldots$

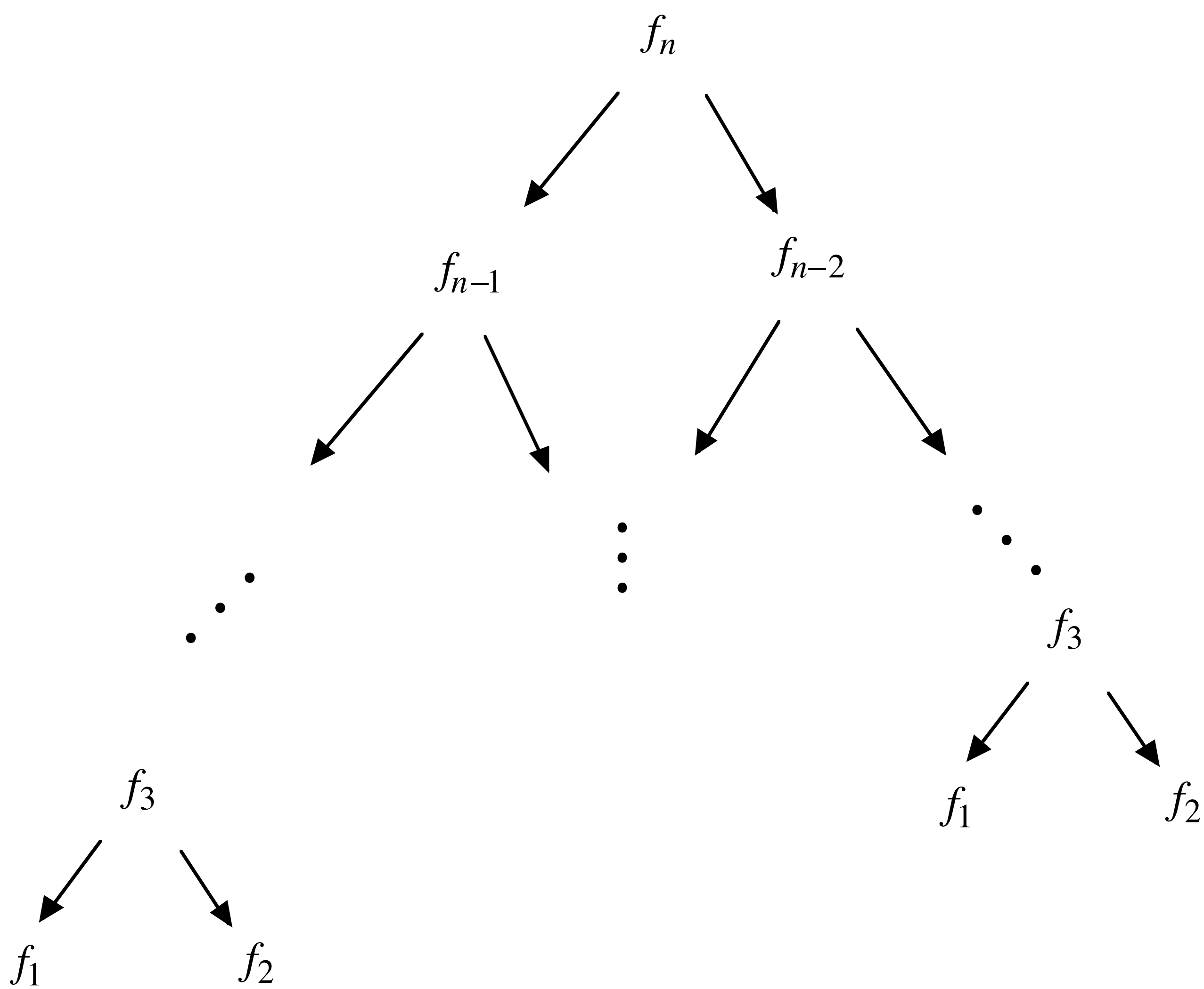
$\Delta f \quad 0\ \ 1\ \ 1\ \ 2\ \ 3\ \ 5\ \ 8\ \ 13\ \ 21\ \ 34$

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

Theorem

There is no polynomial $p(x)$ *for which* $f_n = p(n)$ *for all* n.

Recursive Computation of Fibonacci Numbers



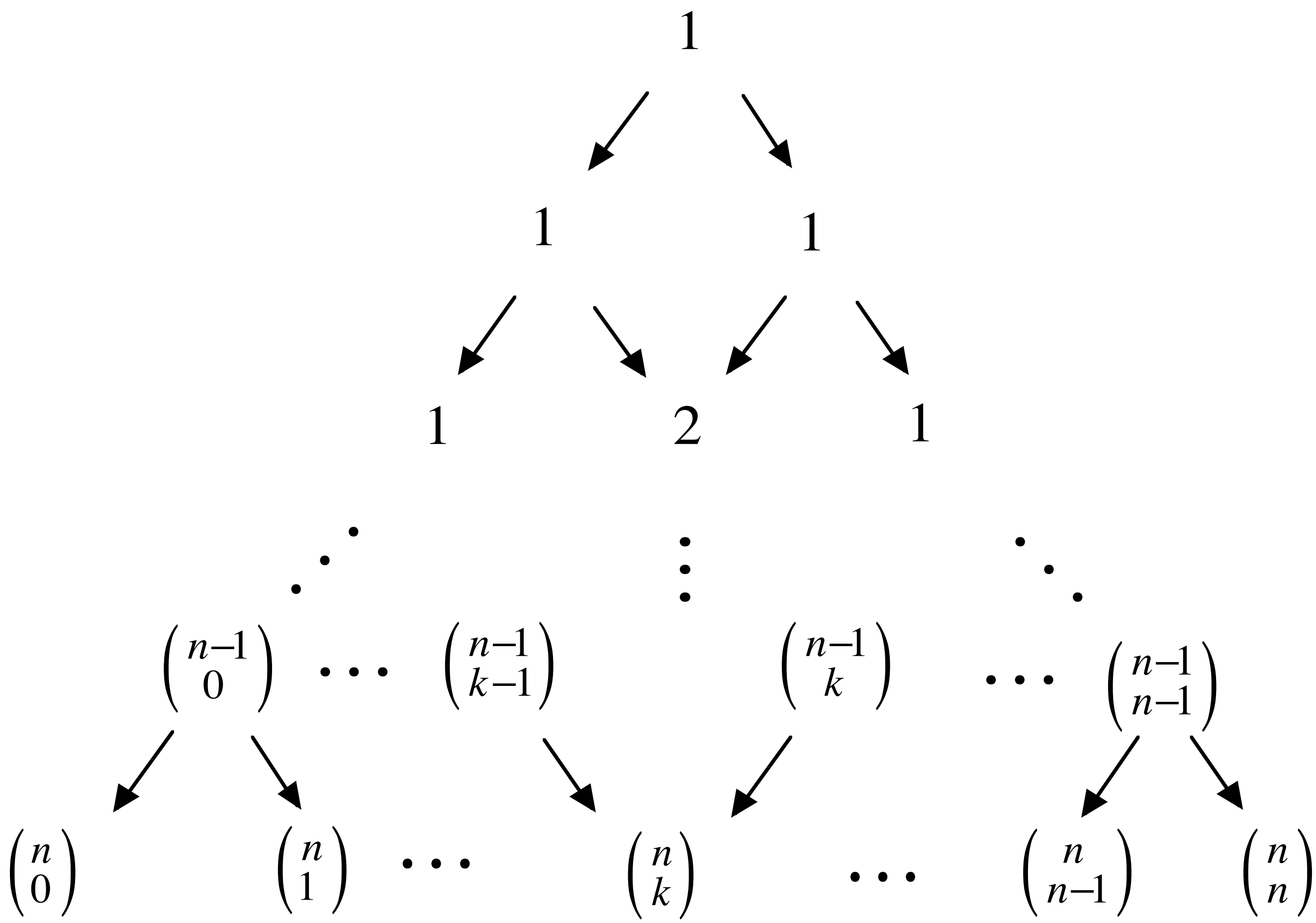
Number of Additions = Exponential

<u>Iterative Computation of Fibonacci Numbers</u>

$$
\begin{tikzcd}
1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 5 \arrow[r] \arrow[d] & \cdots & f_{2n-3} \arrow[r] \arrow[d] & f_{2n-1} \arrow[d] \\
1 \arrow[r] \arrow[ur] & 3 \arrow[r] \arrow[ur] & 8 \arrow[r] & \cdots & f_{2n-2} \arrow[r] \arrow[ur] & f_{2n}
\end{tikzcd}
$$

$$\textit{Number of Additions} = 2(n-2)$$

Iterative Computation of Binomial Coefficients



$$\textit{Number of Additions} = \sum_{k=1}^{n+1} k = \frac{(n+1)(n+2)}{2}$$

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

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

- $D_n \quad = (x_1, y_1), \ldots, (x_n, y_n)$
- $D_n^- = (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$
- $D_n^+ = (x_2, y_2), \ldots, (x_n, y_n)$

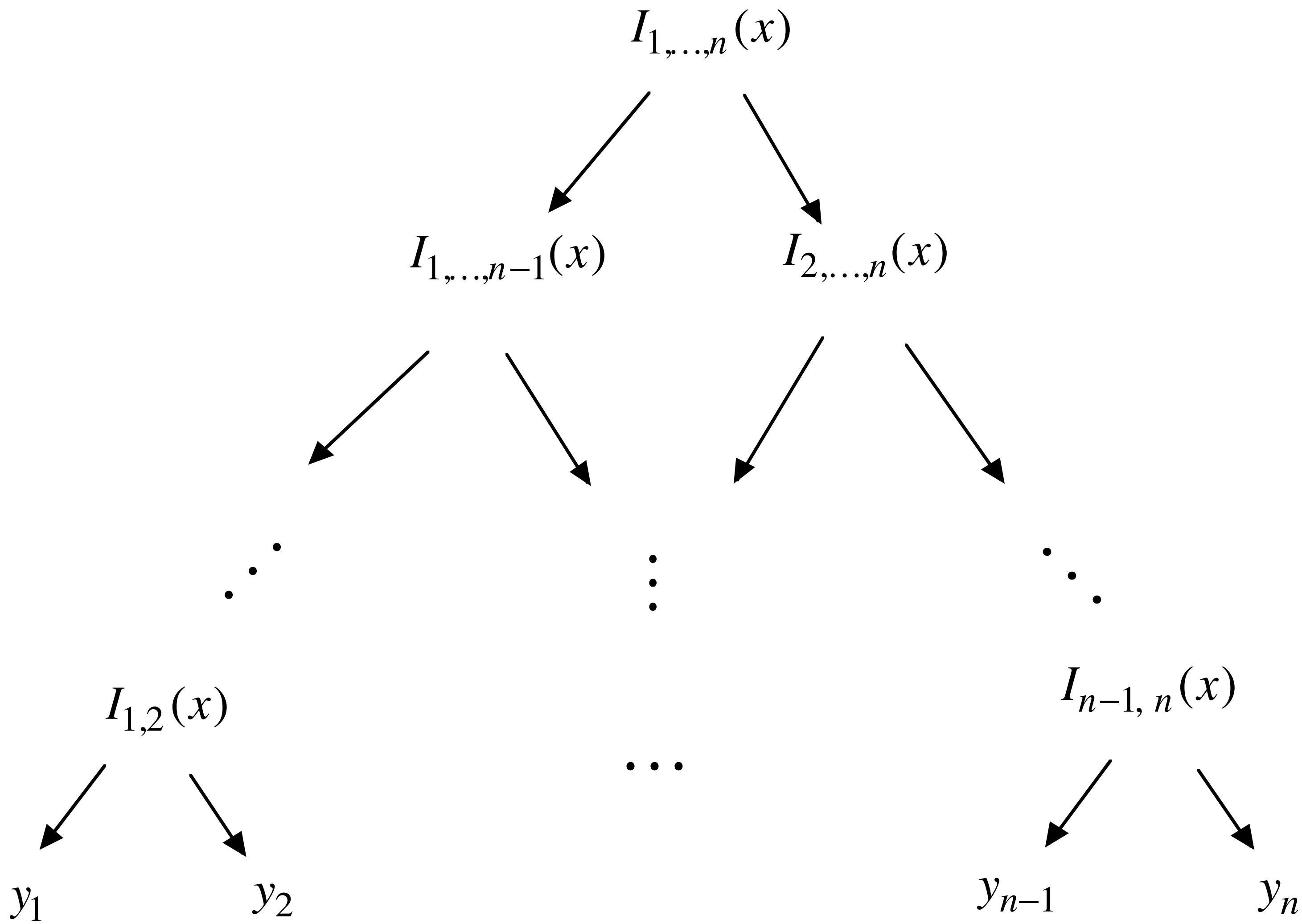
<u>**Polynomial Interpolation and Neville's Algorithm**</u>

Questions

i. How Fast is Neville's Algorithm?

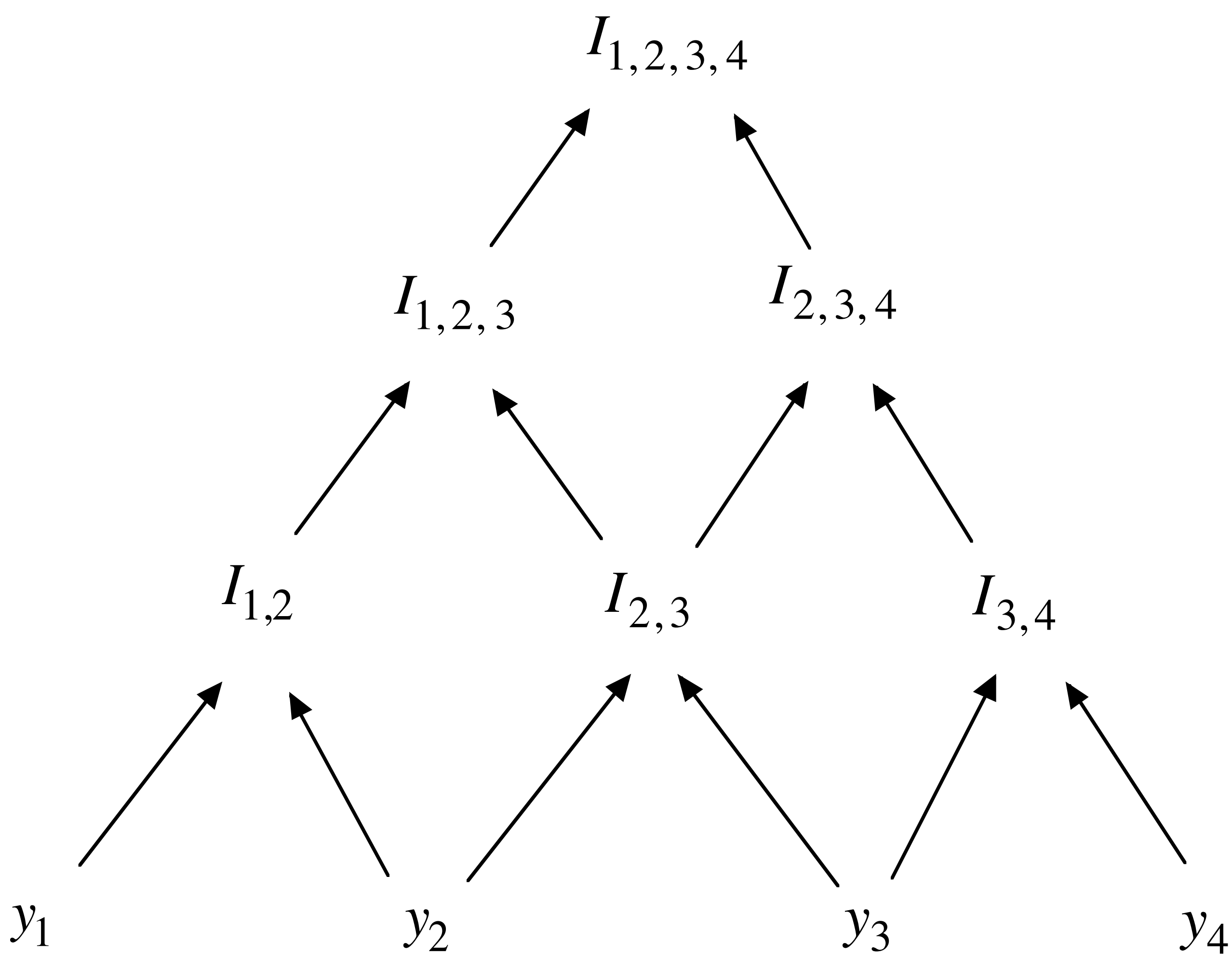
ii. What is the Best Way to Program Neville's Algorithm?

Neville's Algorithm -- Recursive Implementation



$$n-1 \ \textit{Levels} \leftrightarrow 2^n - 2 \ \textit{multiplications}$$

Neville's Algorithm -- Iterative Implementation



$$n-1 \ \ Levels \leftrightarrow \ 2\sum_{k=1}^{n-1} k = (n-1)n \ \ multiplications$$

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

<u>**Tower of Hanoi**</u>

Recursive Solution

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

Number of Moves

- $M(1)=1$
- $M(N+1)=2M(N)+1$

  -- $M(N)=2^N-1$

<u>Program Correctness</u>

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.

<u>Example</u>

Integer Division

- Input: m, n = integers
- Output: q, r = integers (quotient and remainder)

  -- $n = mq + r$ with $0 \le r < m$

Algorithm

$q = 0$

$r = n$

While $r \ge m$

$q = q + 1$

$r = r - m$

Loop Invariant

$n = mq + r$

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:

$$\begin{aligned}(mq+r)_{k+1} &= mq_{k+1} + r_{k+1} \\ &= m(q_k + 1) + r_k - m \\ &= mq_k + r_k && (\textit{inductive hypothesis}) \\ &= n.\end{aligned}$$

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 $y \neq 0$

$r = remainder\ \ of\ \ x\ \ divided\ \ by\ \ y$

(use division algorithm)

$x = y$

$y = r$

Output: x is $GCD(m,n)$

Loop Invariant

$GCD(x, y) = GCD(m, n)$

Proof of Loop Invariance

By induction on the number of iteration of the loop.

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

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:

$$x = yq + r \Rightarrow r = x - yq \Rightarrow GCD(y, r) = GCD(x, y)$$

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 $y \neq 0$ 　　　　　　*Loop Invariant*

$r = remainder\ \ of\ \ x\ \ divided\ \ by\ \ y$ 　　　$GCD(x, y) = GCD(m, n)$

(use division algorithm)

$x = y$

$y = r$

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: $x = GCD(m, n)$.