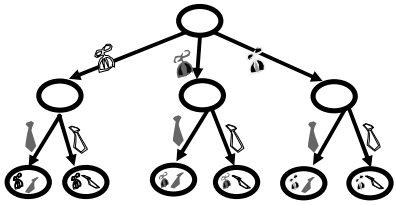
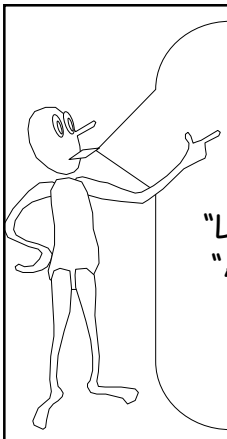
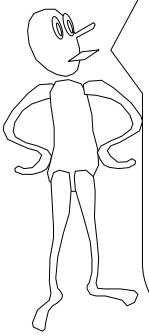
Great Theoretical Ideas In Computer Science		
Anupam Gupta		CS 15-251 Fall 2005
Lecture 3	Sept 6, 2005	Carnegie Mellon University

Induction II: Inductive Pictures





Inductive Proof:
"Standard" Induction
"Least Counter-example"
"All Previous" Induction
and
Invaraints



Theorem? ($k \geq 0$)
$1+2+4+8+\ldots+2^k = 2^{k+1} - 1$

Try it out on small examples:

$2^0 = 2^1 - 1$

$2^0 + 2^1 = 2^2 - 1$

$2^0 + 2^1 + 2^2 = 2^3 - 1$

$S_k \equiv$ "$1+2+4+8+\ldots+2^k = 2^{k+1} - 1$"

Use induction to prove $\forall k \geq 0$, S_k

Establish "Base Case": S_0. We have already checked it.

Establish "Domino Property": $\forall k > 0$, $S_k \Rightarrow S_{k+1}$

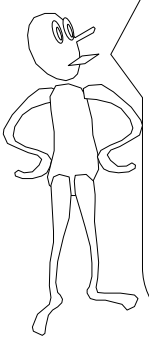
"Inductive Hypothesis" S_k:

$1+2+4+8+\ldots+2^k = 2^{k+1} - 1$

Add 2^{k+1} to both sides:

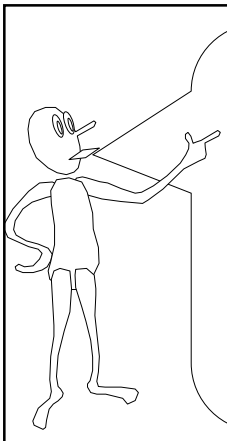
$1+2+4+8+\ldots+2^k + 2^{k+1} = 2^{k+1} + 2^{k+1} - 1$

$1+2+4+8+\ldots+2^k + 2^{k+1} = 2^{k+2} - 1$



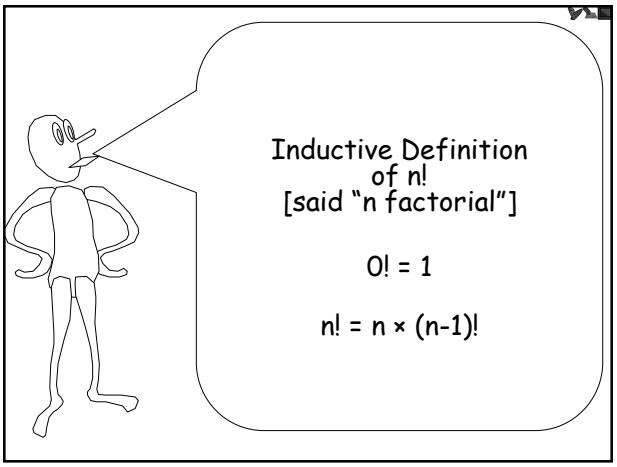
Fundamental lemma of the powers of two:

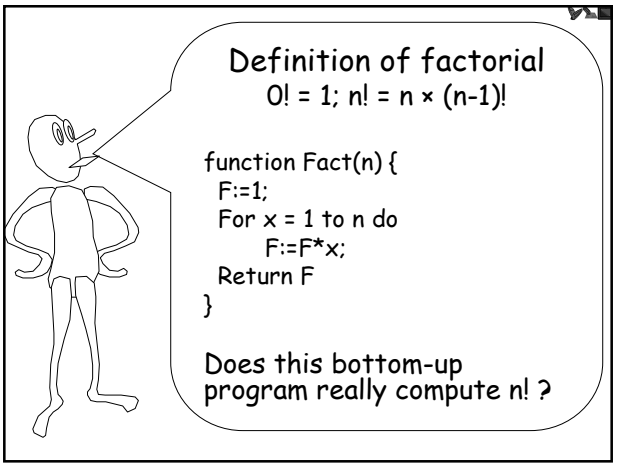
The sum of the first n powers of 2, is one less than the next power of 2.

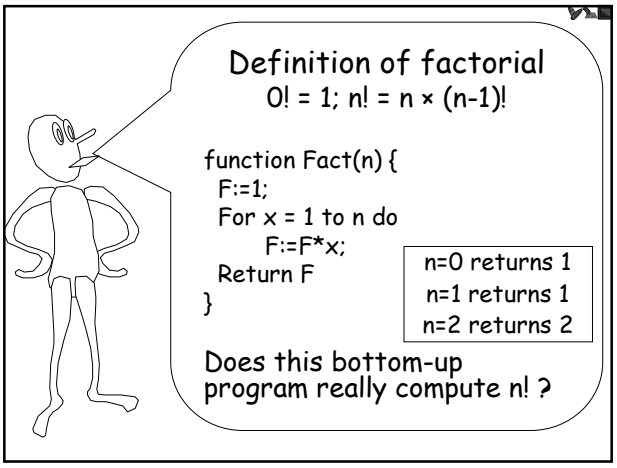


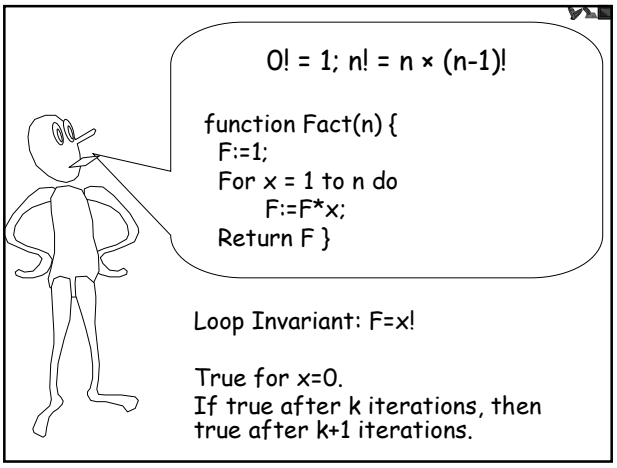
In Lecture #2, we also saw:

Inductive Definitions
of Functions
and
Recurrence Relations









Inductive Definition of T(n)

T(1) = 1

T(n) = 4T(n/2) + n

Notice that T(n) is inductively defined only for positive powers of 2, and undefined on other values.

Inductive Definition of T(n)

T(1) = 1

T(n) = 4T(n/2) + n

Notice that T(n) is inductively defined only for positive powers of 2, and undefined on other values.

T(1)=1 T(2)=6 T(4)=28 T(8)=120

Guess a closed form formula for T(n). Guess G(n)

$G(n) = 2n^2 - n$
Let the domain of G be the powers of two.

Two equivalent functions?

$G(n) = 2n^2 - n$
Let the domain of G be the powers of two.

$T(1) = 1$
$T(n) = 4\ T(n/2) + n$
Domain of T is the powers of two.

Inductive Proof of Equivalence

$G(n) = 2n^2 - n$

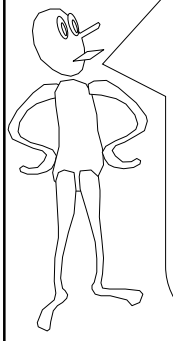
$T(1) = 1$
$T(n) = 4\ T(n/2) + n$

Base: $G(1) = 1$ and $T(1) = 1$

Induction Hypothesis:
$T(x) = G(x)$ for $x < n$

Hence: $T(n/2) = G(n/2) = 2(n/2)^2 - n/2$

$T(n) = 4\ T(n/2) + n$
$= 4\ G(n/2) + n$
$= 4\ [2(n/2)^2 - n/2] + n$
$= 2n^2 - 2n + n$
$= 2n^2 - n$
$= G(n)$



We inductively proved the assertion that G(n) =T(n).

Giving a formula for T with no sums or recurrences is called "solving the recurrence for T".

Solving Recurrences: Guess and Verify

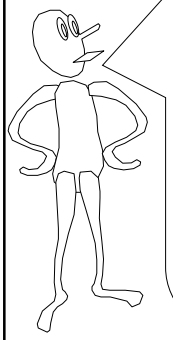
Guess: $G(n) = 2n^2 - n$

$T(1) = 1$
$T(n) = 4\ T(n/2) + n$

Verify: $G(1) = 1$
and $G(n) = 4\ G(n/2) + n$

Similarly: $T(1) = 1$
and $T(n) = 4\ T(n/2) + n$

So $T(n) = G(n)$



That was another view of the same "guess-and-verify" proof of G(n) =T(n).

We showed that G = T at the base case, and that G satisfies the same recurrence relation!

Technique 2
Guess Form and Calculate Coefficients

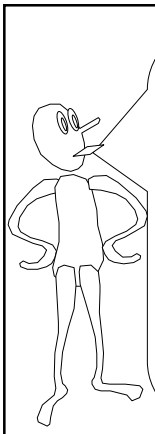
$T(1) = 1$
$T(n) = 4\,T(n/2) + n$

Guess: $T(n) = an^2 + bn + c$
for some a,b,c

Calculate: $T(1) = 1 \Rightarrow a + b + c = 1$

$$T(n) = 4\,T(n/2) + n$$
$$\Rightarrow an^2 + bn + c = 4\,[a(n/2)^2 + b(n/2) + c] + n$$
$$= an^2 + 2bn + 4c + n$$
$$\Rightarrow (b+1)n + 3c = 0$$

Therefore: b=-1 c=0 a=2



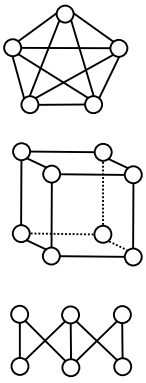
A computer scientist not only deals with numbers, but also with

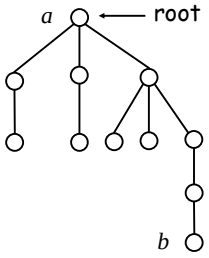
finite strings of symbols

Very visual objects called graphs

And <u>especially</u> the special graphs called trees

Graphs





Definition: Graphs

A graph G = (V,E) consists of
- a finite set V of vertices (also called "nodes"), and
- a finite set E of edges.

Each edge is a set {a, b} of two different vertices.

n.b. A graph may not have self loops or multiple edges (unless mentioned otherwise)

Definition: Directed Graphs

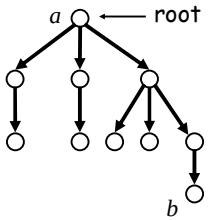
A graph G = (V,E) consists of
- a finite set V of vertices (nodes) and
- a finite set E of edges.

Each edge is an <u>ordered</u> pair <a,b> of two different vertices.

n.b. A directed graph may not have self loops or multiple edges (unless mentioned otherwise).

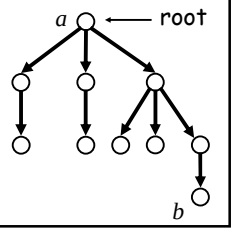
Definition: Tree

A tree is a directed graph with one special node called the root and the property that each node must a <u>unique path</u> from the root to itself.



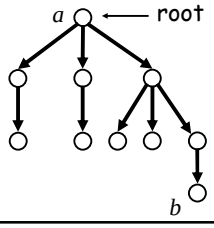
Terminology: Tree

- Child: If <u,v>∈E, we say v is a child of u
- Parent: If <u,v>∈E, we say u is the parent of v
- Leaf: If u has no children, we say u is leaf.
- Siblings: If u and v have the same parent, they are siblings.



Terminology: Tree

- Descendants of u:
  The set of nodes reachable from u (including u).
- Sub-tree rooted at u:
  Descendants of u and all the edges between them.
  (u is designated as the root of this tree.)

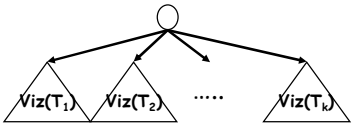


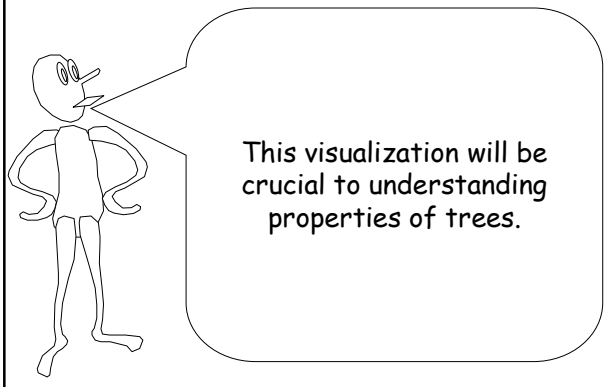
Classical Visualizations of Trees

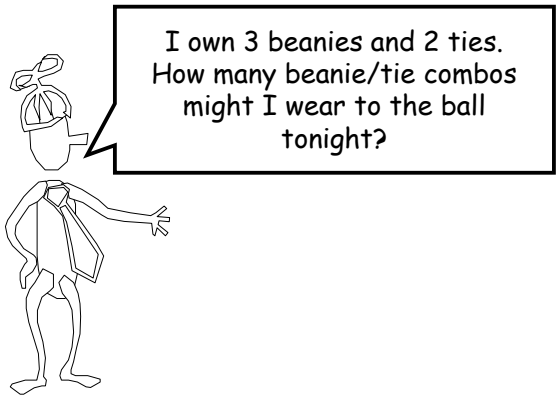
Here's the inductive rule:

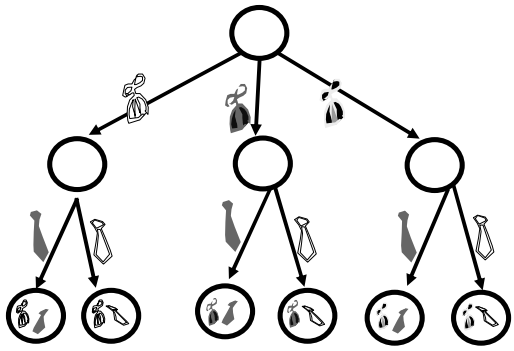
If G is a single node
Viz(G) = ◯

If G consists of root r with sub-trees T_1, T_2, …, T_k
Viz(G) =

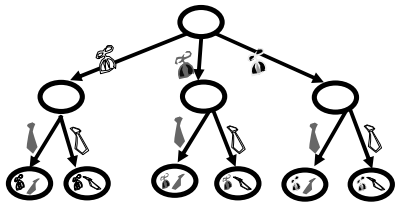








Choice Tree

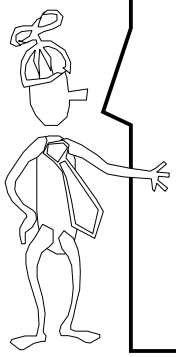


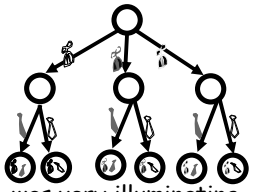
A choice tree is a tree with an object called a "choice" associated with each edge and a label called an "outcome" on each leaf.

Definition: Labeled Trees

A tree "node labeled by S" is
a tree T = (V,E) with
an associated function Node-Label: $V \rightarrow S$

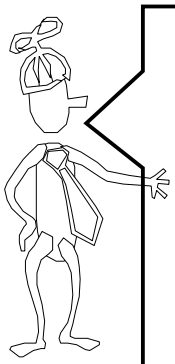
A tree "edge labeled by S" is
a tree T = (V,E) with
an associated function Edge-Label: $E \rightarrow S$





was very illuminating.

Let's do something similar to illuminate the nature of T(1)=1; T(n) = 4T(n/2) + n



T(1)=1; T(n)= 4T(n/2) + n

For each n (power of 2), we will define a tree V(n) that is node-labeled by numbers.

This tree V(n) will give us an incisive picture of the recurrence relation T(n).

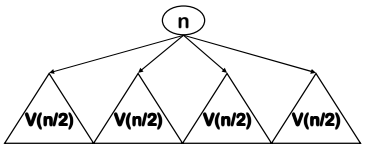
Inductive Definition Of Labeled Tree V(n)

T(1) = 1

V(1) = 1

T(n) = n + 4 T(n/2)

V(n) =



Inductive Definition Of Labeled Tree V(n)

T(1) = 1

V(1) = 1

T(1) = 1
T(n) = n + 4 T(n/2)

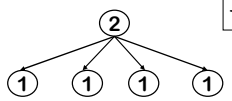
V(2) =

Inductive Definition Of Labeled Tree V(n)

T(2) = 6

T(1) = 1
T(n) = n + 4 T(n/2)

V(2) =



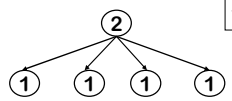
V(4) =

Inductive Definition Of Labeled Tree V(n)

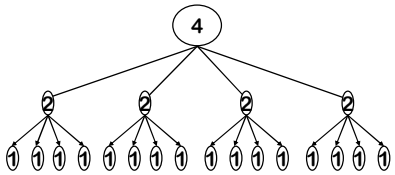
T(2) = 6

T(1) = 1
T(n) = n + 4 T(n/2)

V(2) =



V(4) =

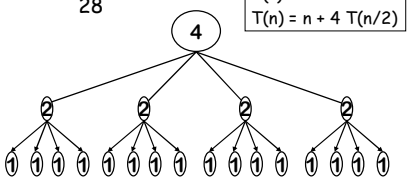


Inductive Definition Of Labeled Tree V(n)

T(4) = 28

T(1) = 1
T(n) = n + 4 T(n/2)

V(4) =

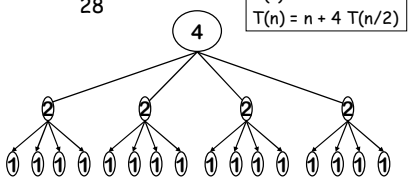


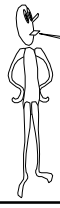
Inductive Definition Of Labeled Tree V(n)

T(4) = 28

T(1) = 1
T(n) = n + 4 T(n/2)

V(4) =





When we sum the node labels on V(n), we get back T(n)

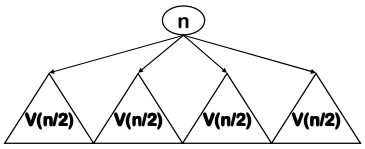
Can prove Invariant: T(n) = (sum of labels in V(n))

T(1) = 1

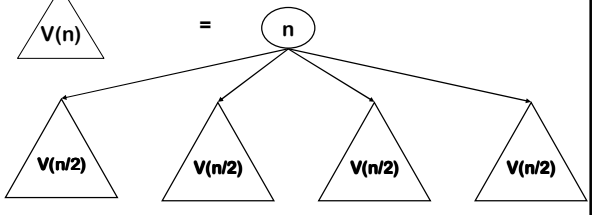
V(1) =

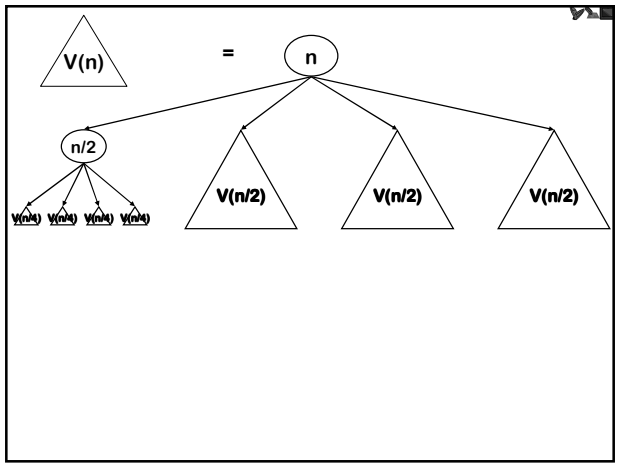
T(n) = n + 4 T(n/2)

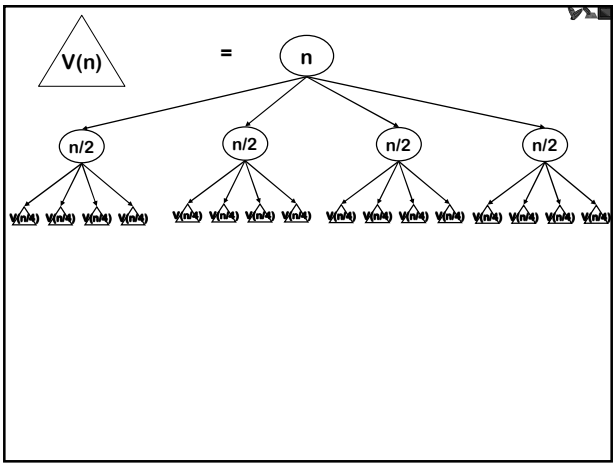
V(n) =

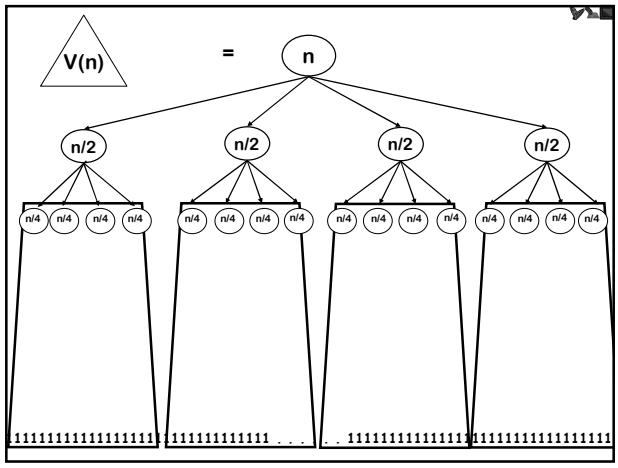


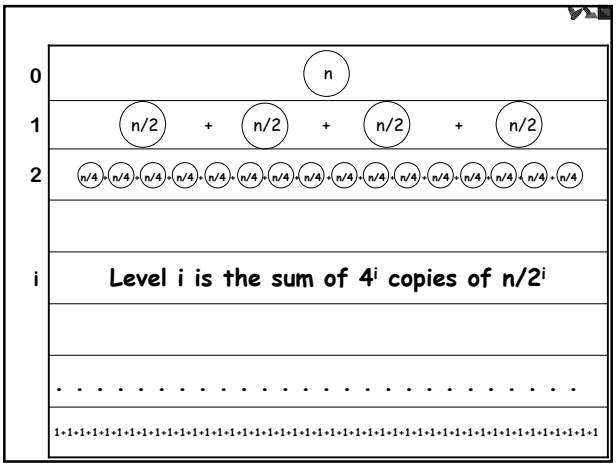
V(n) =

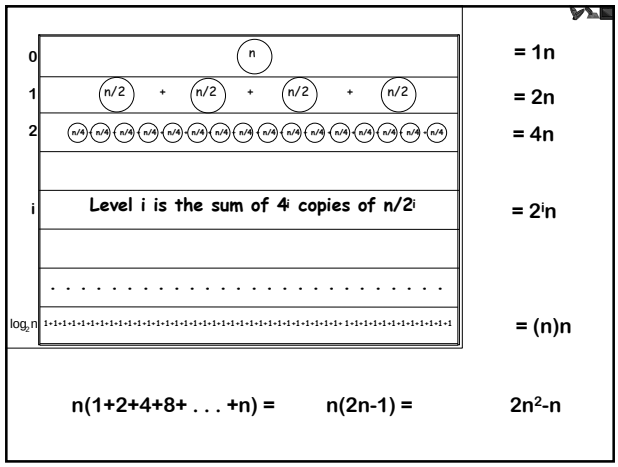


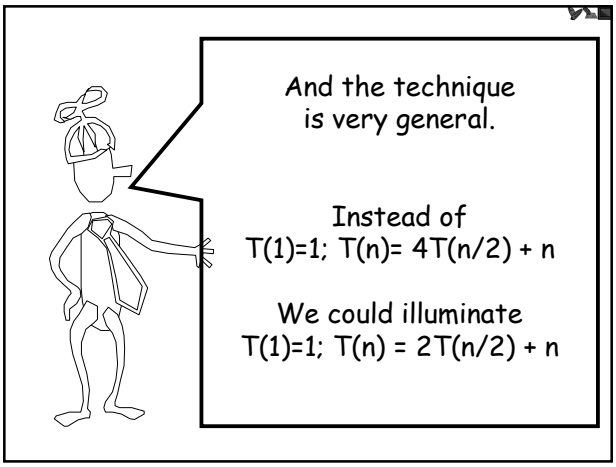

V(n)
=
n
n/2
V(n/2)
V(n/2)
V(n/2)


V(n)
=
n
n/2
n/2
n/2
n/2


V(n)
=
n
n/2
n/2
n/2
n/2
n/4


0
1
2
i
n
n/2 + n/2 + n/2 + n/2
Level i is the sum of 4^i copies of n/2^i
1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1


0
1
2
i
log2n
n
n/2 + n/2 + n/2 + n/2
Level i is the sum of 4^i copies of n/2^i
= 1n
= 2n
= 4n
= 2^in
= (n)n
n(1+2+4+8+ . . . +n) =
n(2n-1) =
2n^2-n


And the technique is very general.
Instead of T(1)=1; T(n)= 4T(n/2) + n
We could illuminate T(1)=1; T(n) = 2T(n/2) + n

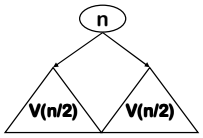
New recurrence: T(1)=1; T(n) = 2T(n/2) + n

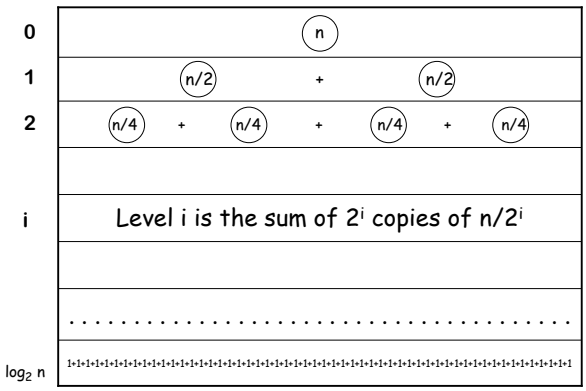
T(1) = 1

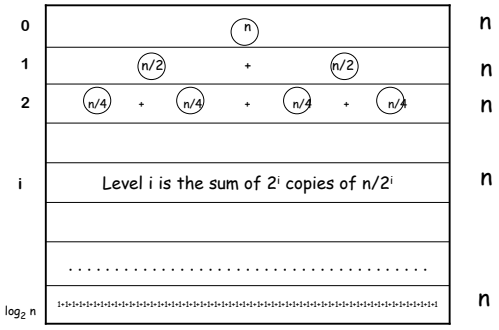
= 1

T(n) = n + 2 T(n/2)

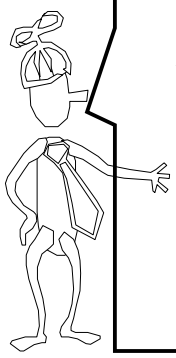
=







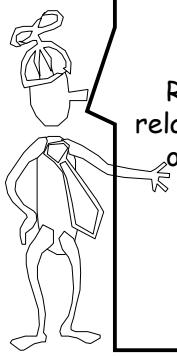
n(1+1+1+…+1) = n $\log_2$ n



T(1)=1; T(n) = 2T(n/2) + n

has closed form
T(n) = n $\log_2$(n)

where n is a power of 2



Representing a recurrence relation as a labeled tree is one of the basics tools you will have to put recurrence relations in closed form.

The Lindenmayer Game

Alphabet: Σ = {a,b}

Start word: a

Sub(a) = ab Sub(b) = a

NEXT(w_1 w_2 … w_n) =
Sub(w_1) Sub(w_2) … Sub(w_n)

The Lindenmayer Game

Sub(a) = ab Sub(b) = a
NEXT(w_1 w_2 … w_n) = Sub(w_1) Sub(w_2) … Sub(w_n)

Time 1: a
Time 2: ab
Time 3: aba
Time 4: abaab
Time 5: abaababa

The Lindenmayer Game

Sub(a) = ab Sub(b) = a
NEXT(w_1 w_2 … w_n) = Sub(w_1) Sub(w_2) … Sub(w_n)

Time 1: a
Time 2: ab
Time 3: aba
Time 4: abaab
Time 5: abaababa

How long are the strings as a function of time?

Aristid Lindenmayer (1925-1989)

1968 Invents L-systems in Theoretical Botany

Time 1: a
Time 2: ab
Time 3: aba
Time 4: abaab
Time 5: abaababa

FIBONACCI(n) cells at time n

The Koch Game

Alphabet: Σ = { F, +, - }

Start word: F

Sub(F) = F+F--F+F
Sub(+) = +
Sub(-)=-

NEXT(w_1 w_2 … w_n) = Sub(w_1) Sub(w_2) … Sub(w_n)

The Koch Game

Gen 0: F
Gen 1: F+F--F+F
Gen 2: F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

F+F--F+F + F+F--F+F - - F+F--F+F + F+F--F+F

The Koch Game

Gen 0: F
Gen 1: F+F--F+F
Gen 2: F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

Visual representation:

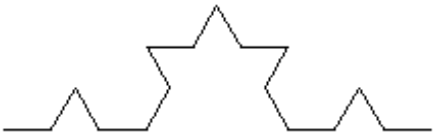
F draw forward one unit
\+ turn 60 degree left
\- turn 60 degrees right.

The Koch Game

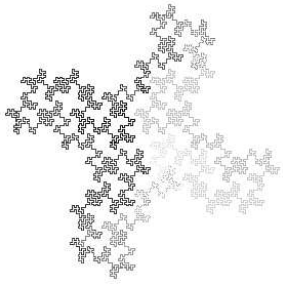
F+F--F+F

The Koch Game

F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

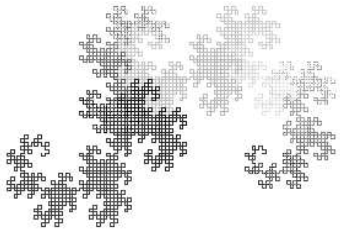


Koch Curve



Dragon Game

Sub(X) =X+YF+	Sub(Y) = -FX-Y

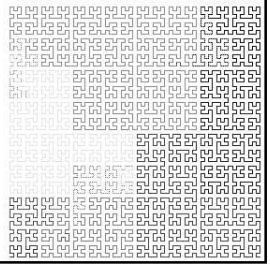


Dragon Curve

Hilbert Game

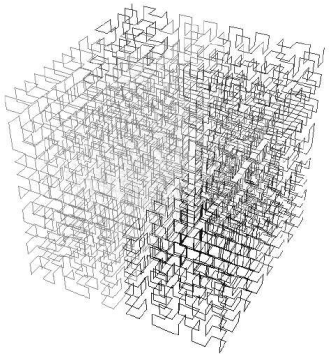
Sub(L)= +RF-LFL-FR+
Sub(R)= -LF+RFR+FL-

Hilbert Curve

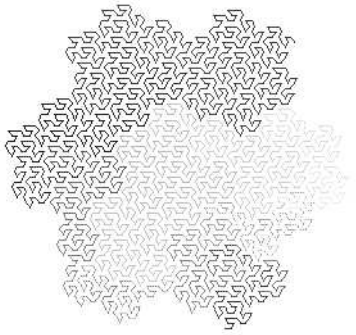


Note: Make 90 degree turns instead of 60 degrees.

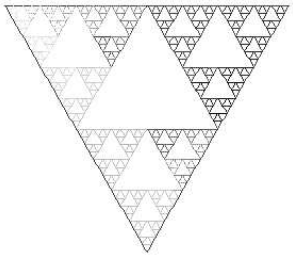
Hilbert's Space Filling Curve



Peano-Gossamer Curve



Sierpinski Triangle



Lindenmayer (1968)

Sub(F) = F[-F]F[+F][F]

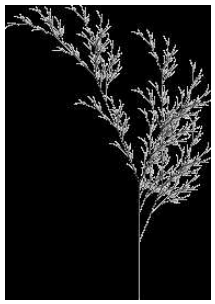
Interpret the stuff inside brackets as a branch.

Lindenmayer 1968

Inductive Leaf

Start at X
Sub(X) = F-[[X]+X]+F[+FX]-X
Sub(F) = FF

Angle=22.5



Study Bee

Recap:
- Inductive Proofs
- Invariants
- Inductive Definitions of Functions

Summing Powers of 2

Solving Recurrence Relations
- Guess and Verify
- Visualizing recurrences as labeled trees

Definitions
- Factorial function
- Graphs (Undirected/Directed)
- Trees and Labelings