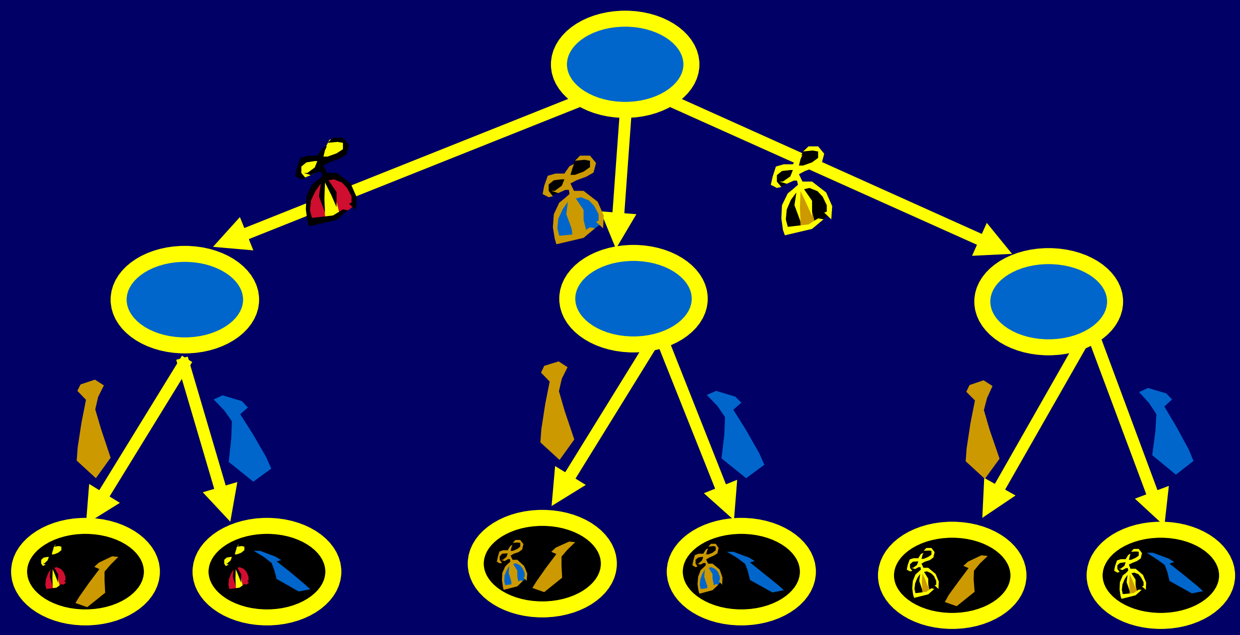
Great Theoretical Ideas In Computer Science

Anupam Gupta CS 15-251 Fall 2006

Lecture 3 Sept 5, 2006 Carnegie Mellon University

Induction II: Inductive Pictures



Inductive Proof:

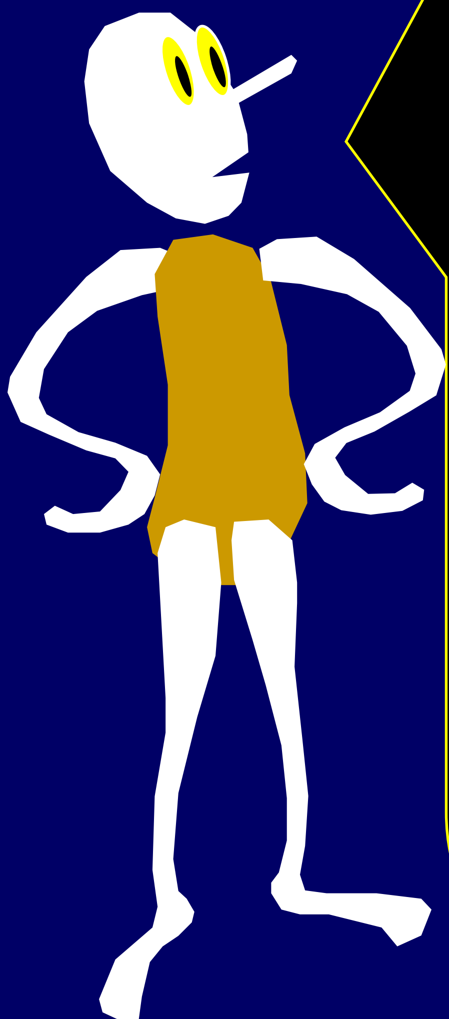
"Standard" Induction

"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

Base Case: S_k for $k=0$ $\quad (S_0)$

true (see last page)

Induction Hypothesis: Assume S_k is true

$$1+2+\cdots+2^k = 2^{k+1}-1$$

hence

$$\begin{aligned}
1+2+\cdots+2^k+2^{k+1} &= (1+2+\cdots+2^k)+2^{k+1} \\
&= (2^{k+1}-1)+2^{k+1} = 2\cdot 2^{k+1}-1 \\
&= 2^{k+2}-1
\end{aligned}$$

$\therefore S_{k+1}$ is true ☺

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

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

(1) ~~1st~~ 0th domino fell

(2) $(S_k \Rightarrow S_{k+1})$ $\forall k \geq 0$

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

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

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

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

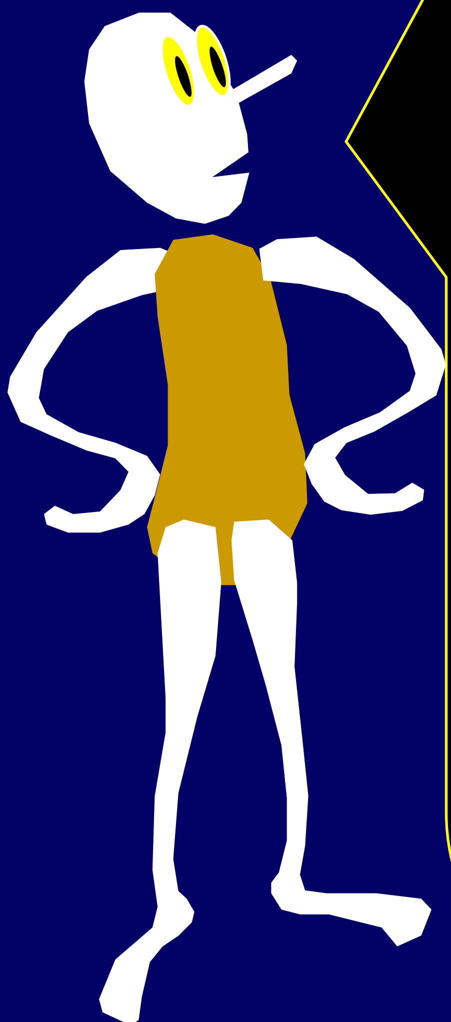
"Inductive Hypothesis" S_k:

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

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

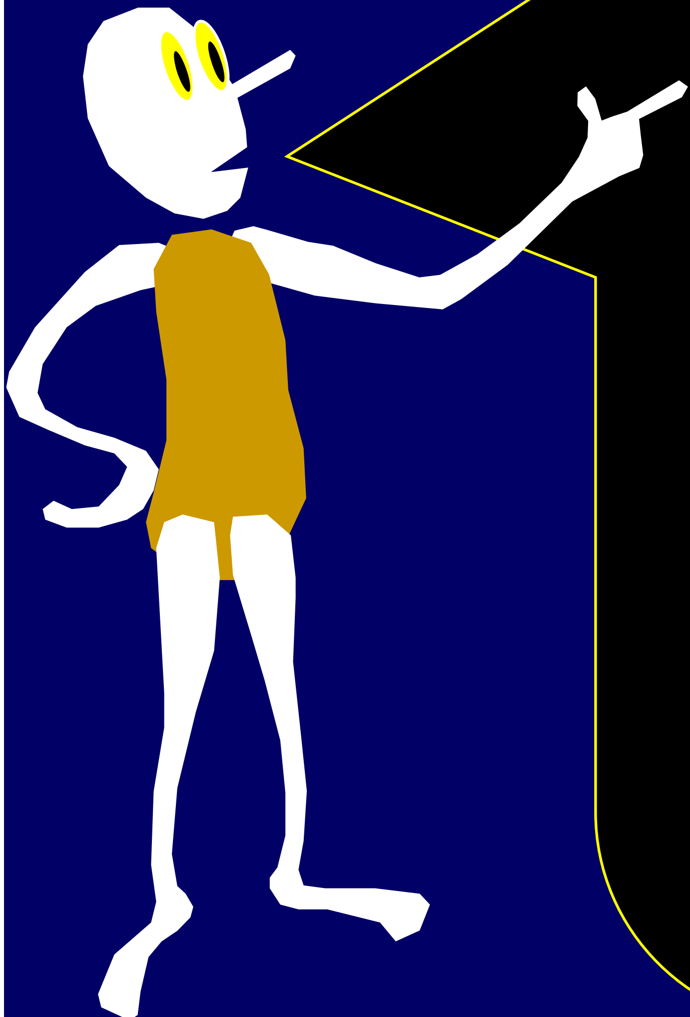
$1+2+4+8+\ldots+2^k + 2^{k+1} = 2^{k+1} + 2^{k+1} - 1$

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



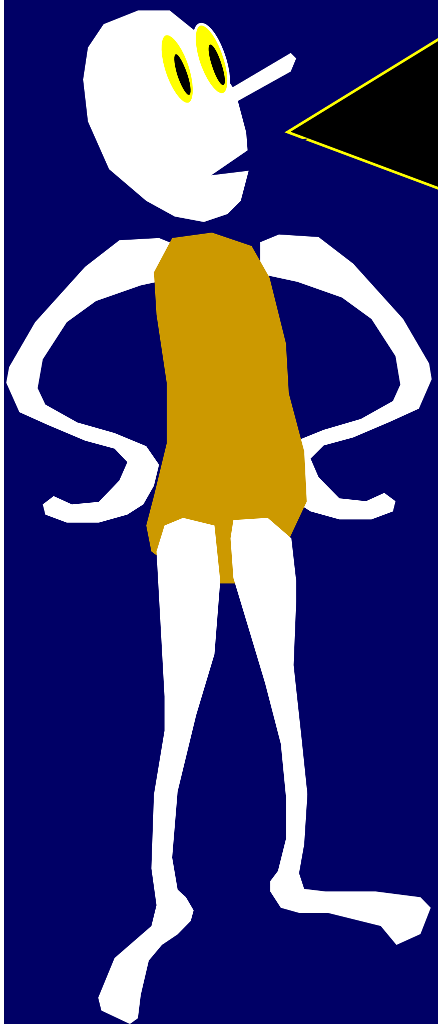
Fundamental lemma of the powers of two:

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



In Lecture #2,
we also saw:

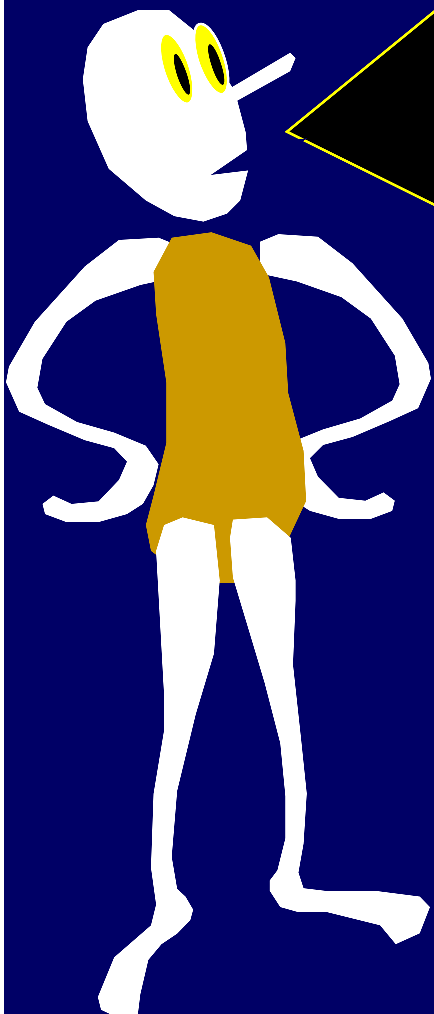
Inductive Definitions
of Functions
and
Recurrence Relations



Inductive Definition of n! [said "n factorial"]

$$0! = 1$$

$$n! = n \times (n-1)!$$



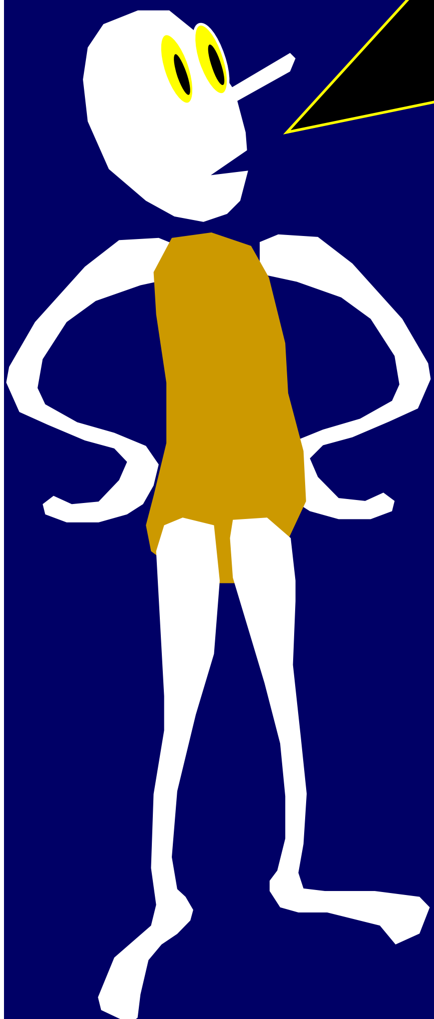
Definition of factorial

$0! = 1;\ n! = n \times (n-1)!$

```
function Fact(n) {
  F:=1;
  For x = 1 to n do
       F:=F*x;
  Return F
}
```

n=0 returns 1
n=1 returns 1
n=2 returns 2

Does this bottom-up program really compute n! ?



$0! = 1;\ n! = n \times (n-1)!$

```
function Fact(n) {
  F:=1;
  For x = 1 to n do {
        F:=F*x;
        // Loop Invariant: F=x! here}
  Return F
  }
```

Loop Invariant: F=x!

True for x=0.

If true after k iterations, then
true after k+1 iterations.

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.

$G(1)=1$ $G(2)=6$ $G(4)=28$ $G(8)=120$

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

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

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

Inductive Proof of Equivalence

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

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

Inductive Proof of Equivalence

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

$T(1) = 1$

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

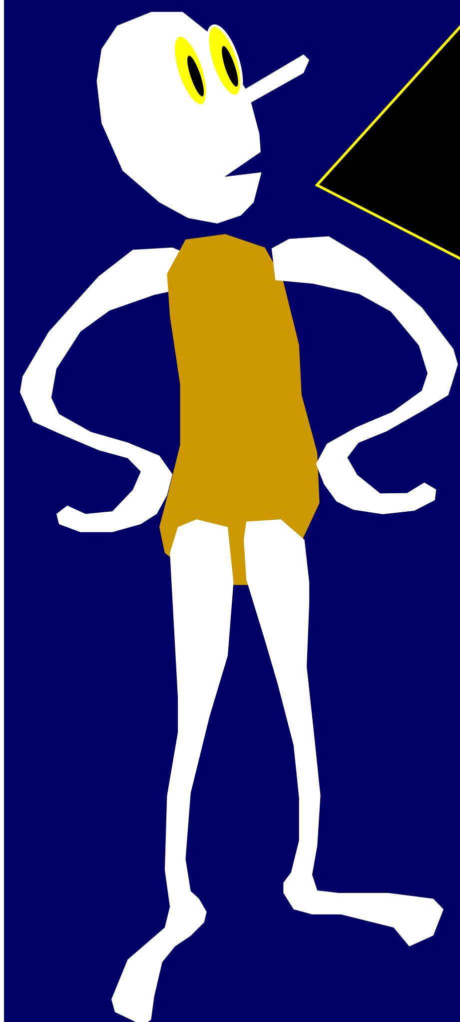
Base: $G(1) = 1$ and $T(1) = 1$

Induction Hypothesis:

$T(x) = G(x)$ for $x < n$, x being a power of 2.

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

$T(n)$	$= 4\,T(n/2) + n$	[by definition of T(n)]
	$= 4\,G(n/2) + n$	[by I.H.]
	$= 4\,[2(n/2)^2 - n/2] + n$	[by definition of G(n/2)]
	$= 2n^2 - 2n + n$	
	$= 2n^2 - n$	
	$= G(n)$	[by definition of G(n)]



We inductively proved the assertion that $\forall\, n \geq 1,\ T(n) = G(n)$.

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

Solving Recurrences: Guess and Verify

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

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

(I) $G(1) = 2 \cdot 1^2 - 1 = 1$

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

$$= 4\left[2\left(\frac{n}{2}\right)^2 - \frac{n}{2}\right] + 2\,\frac{n}{2}$$

$$G(n) = 4\,G\left(\frac{n}{2}\right) + n$$

Solving Recurrences: Guess and Verify

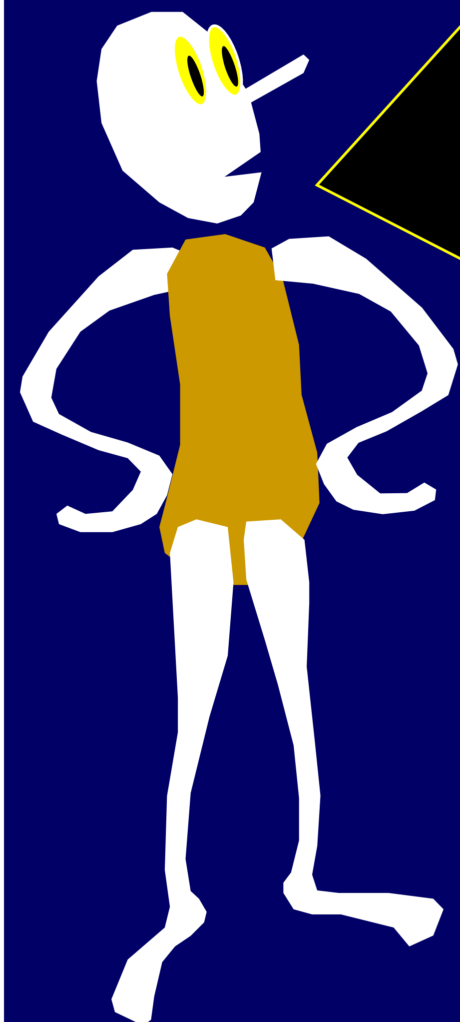
$T(1) = 1$
$T(n) = 4\ T(n/2) + n$

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

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

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

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

$T(1) = 1$

$\Rightarrow a \cdot 1^2 + b \cdot 1 + c = 1$

$\Rightarrow \boxed{a + b + c = 1}$

Technique 2
Guess Form and Calculate Coefficients

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

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

$$\cancel{an^2} + bn + c = 4\left[a\left(\tfrac{n}{2}\right)^2 + b\left(\tfrac{n}{2}\right) + c\right] + n$$

$$= \cancel{an^2} + 2bn + 4c + n$$

$$\Rightarrow \quad bn + n + 3c = 0 \qquad \forall n$$

$$\Rightarrow \quad (b+1)n + 3c = 0 \qquad \forall n \text{ power of } 2.$$

$$\Rightarrow \quad b+1 = 0 \quad , \quad c = 0$$

$$\Rightarrow \quad b = -1, \qquad c = 0$$

Technique 2
Guess Form and Calculate Coefficients

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

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

$a+b+c = 1$

$b = -1$

$c = 0 \qquad \Rightarrow \quad a = 2$

$T(n) = 2n^2 + (-1).n + 0$

$= 2n^2 - n.$

Technique 2
Guess Form and Calculate Coefficients

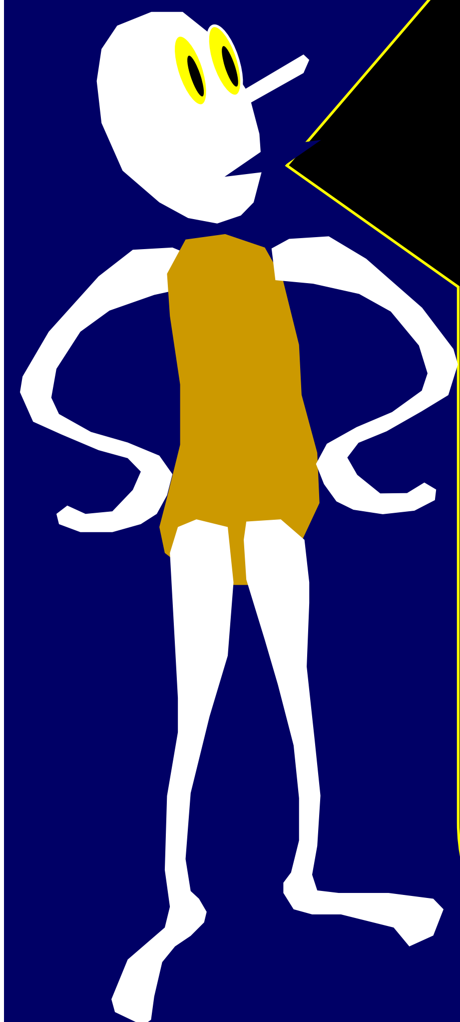
$T(1) = 1$
$T(n) = 4\ T(n/2) + n$

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

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

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

Therefore: $b=-1 \quad c=0 \quad a=2$



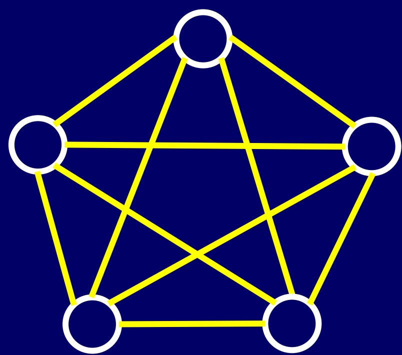
A computer scientist not only deals with numbers, but also with

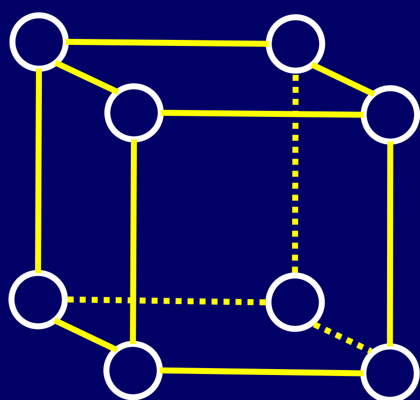
finite strings of symbols

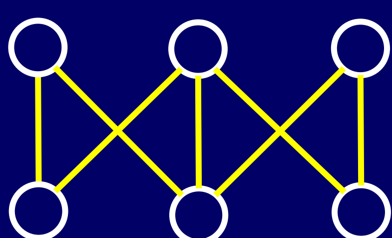
Very visual objects called graphs

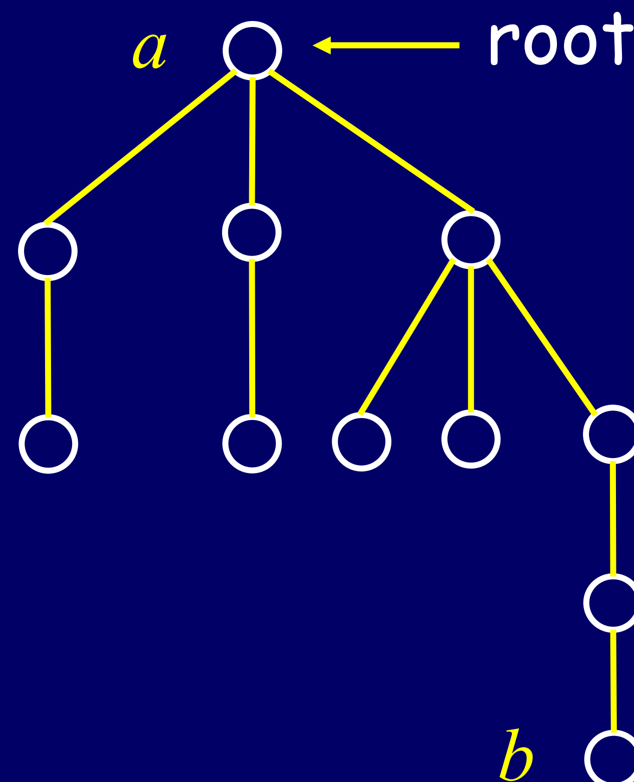
And <u>especially</u> the special graphs called trees

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)

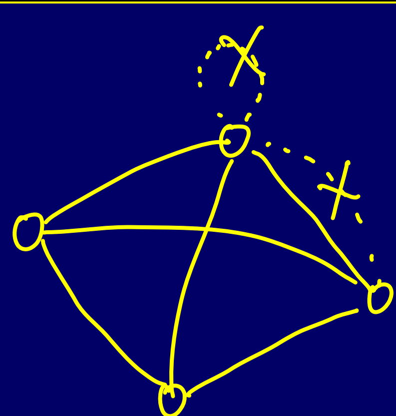
Visualization: graphs

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)



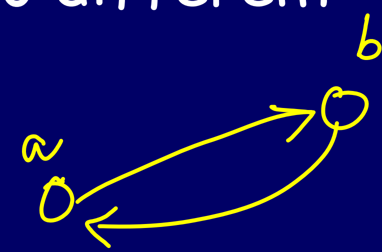
Any graph on n nodes has at most $\frac{n(n-1)}{2}$ edges.

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 ordered pair <a,b> of two different vertices.



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

Visualization: Directed graphs

A graph G = (V,E) consists of

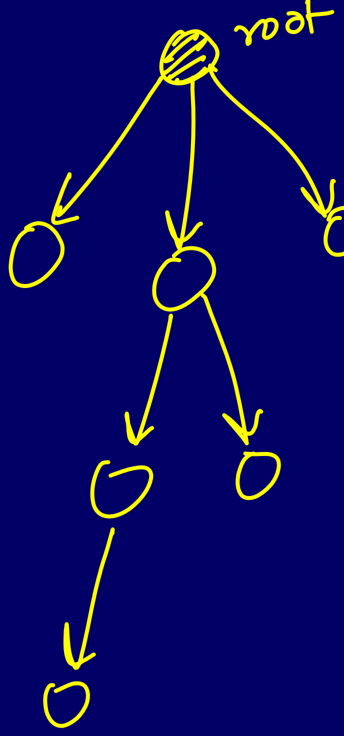
- a finite set V of vertices (nodes) and
- a finite set E of edges.

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).

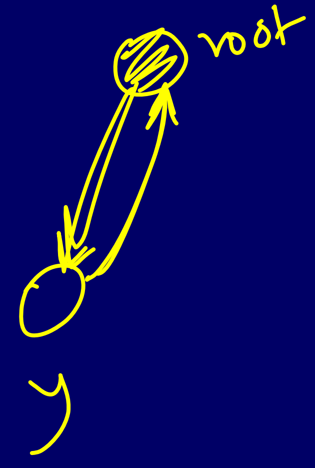
Definition: Rooted Tree

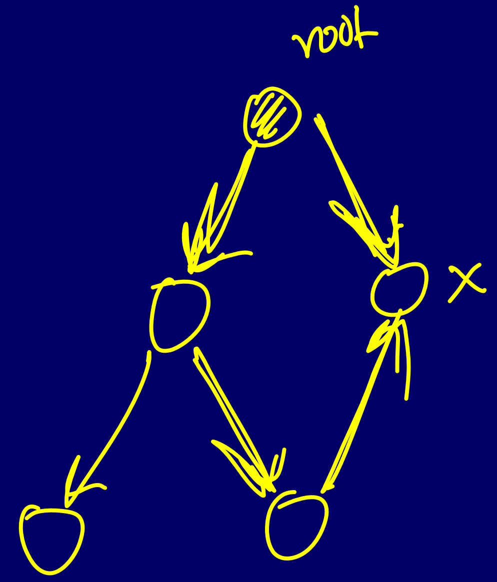
A rooted tree is a directed graph with one special node called the root and the property that each node has a unique path from the root to itself.



Are these trees?

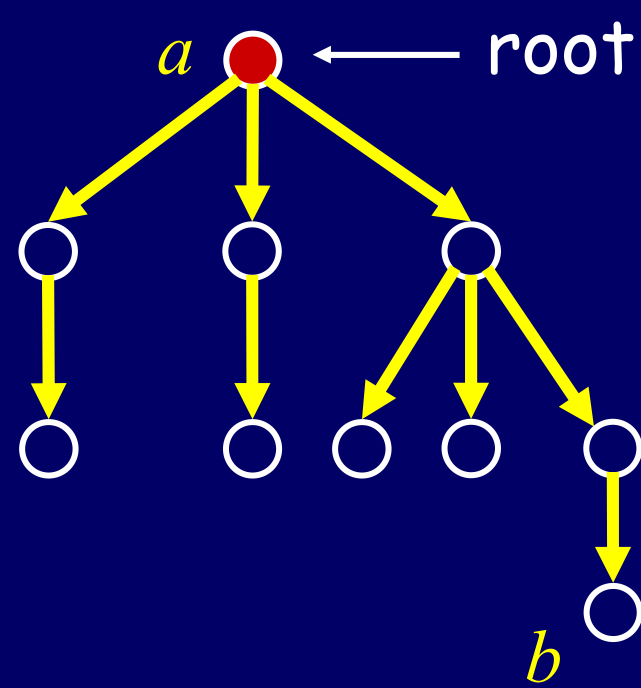
A rooted tree is a directed graph with one special node called the root and the property that each node has a <u>unique path</u> from the root to itself.





Definition: Rooted Tree

A rooted tree is a directed graph with one special node called the root and the property that each node has a unique path from the root to itself.



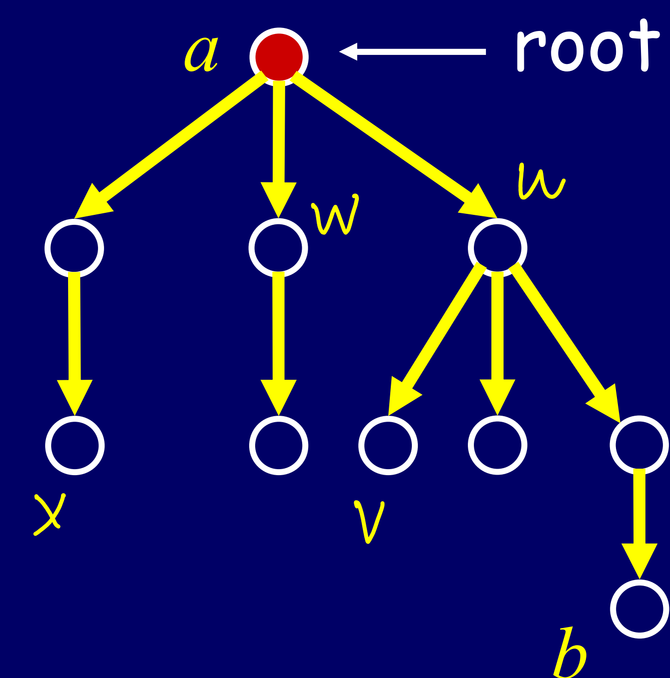
Terminology: Tree

Child: If $\langle u,v \rangle \in E$, we say v is a child of u

Parent: If $\langle u,v \rangle \in E$, we say u is the parent of v

Leaf: If x has no children, we say x is leaf.

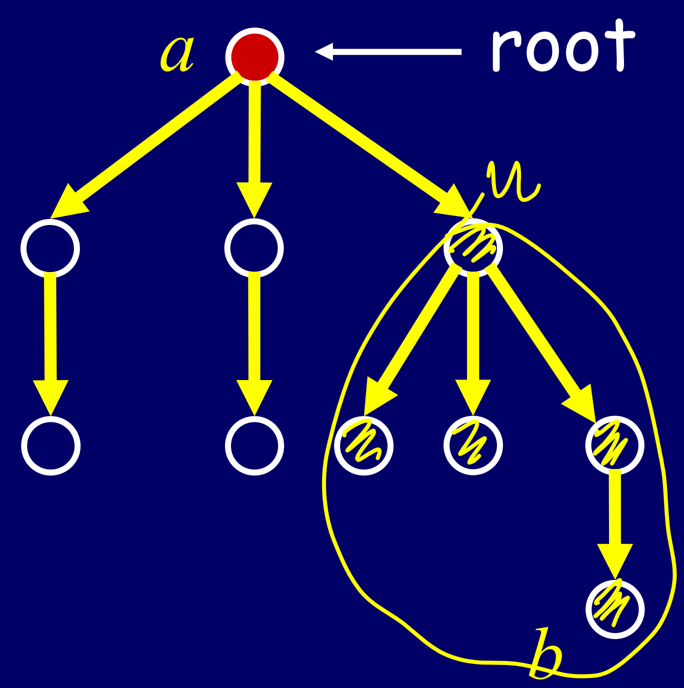
Siblings: If u and w have the same parent, they are siblings.



Terminology: Tree

Descendants of u:
The set of nodes reachable from u (including u).

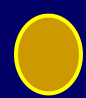
Sub-tree rooted at u:
Descendants of u and all the edges between them.
(u is designated as the root of this tree.)



Classical Visualizations of Trees

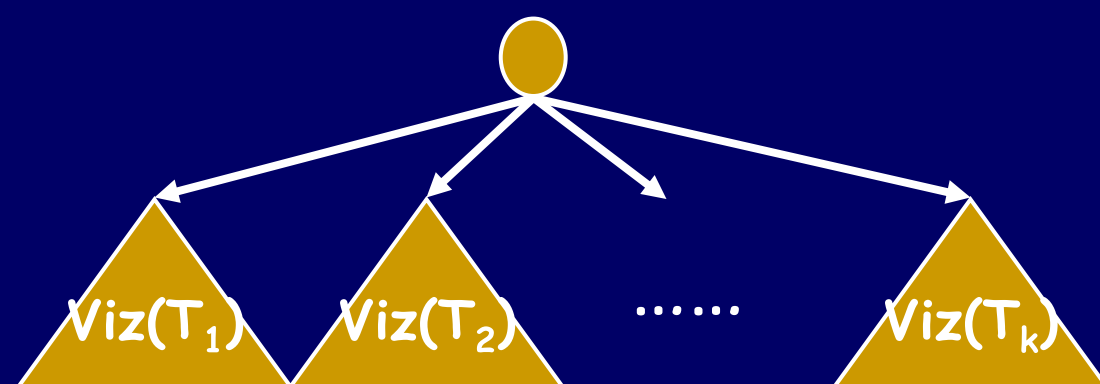
<u>Here's the inductive rule:</u>

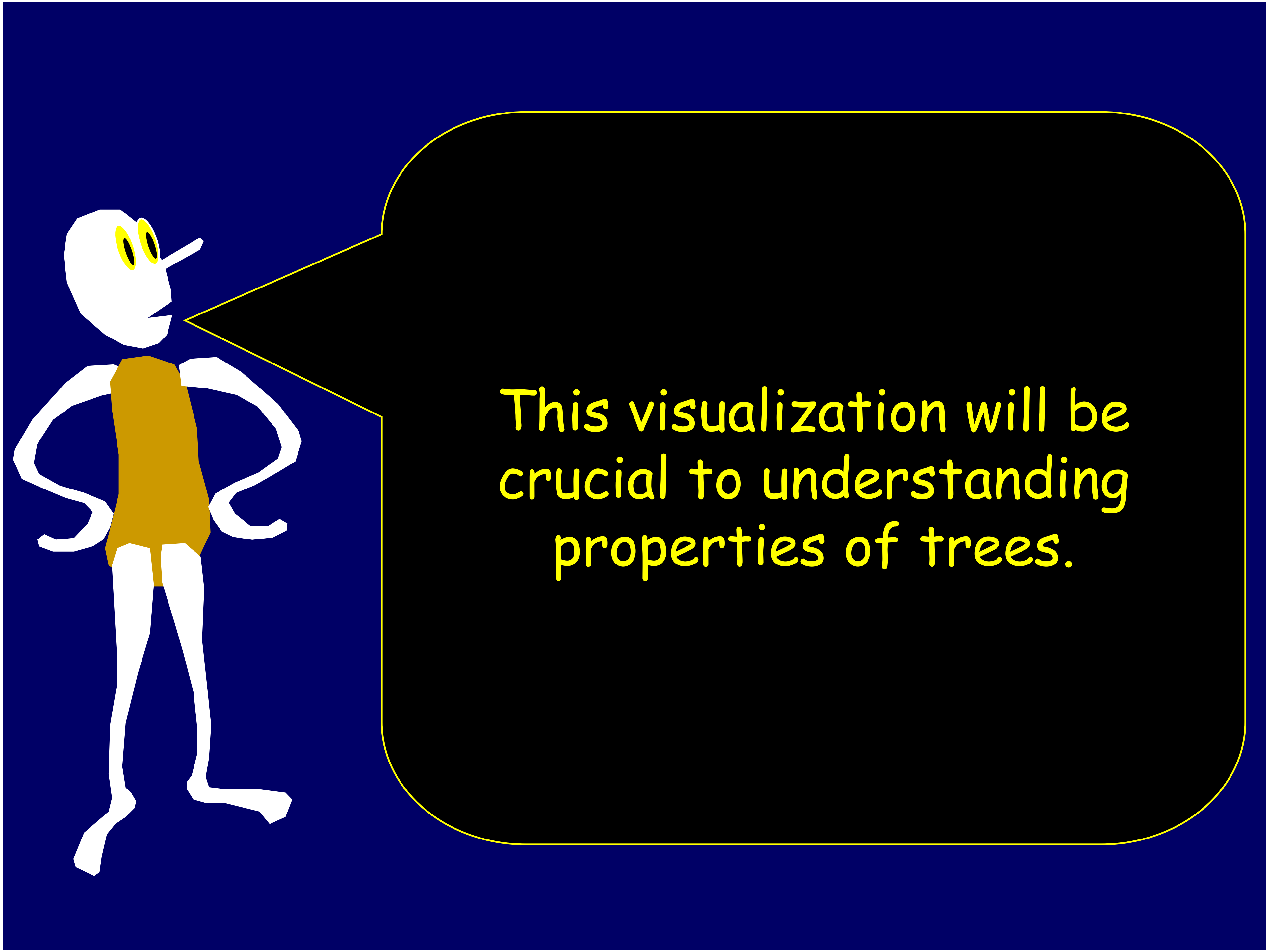
If G is a single node

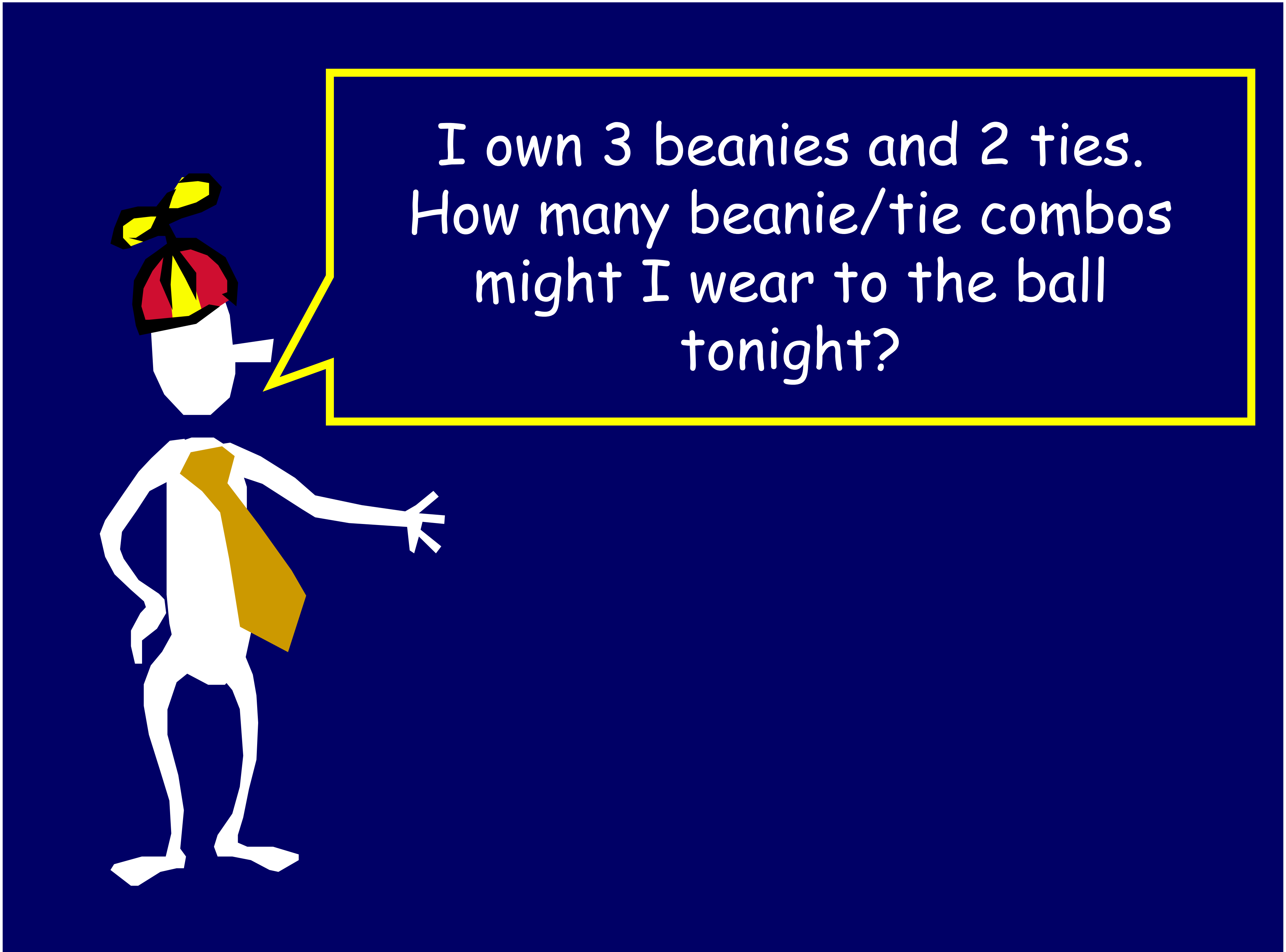
$Viz(G) =$ 

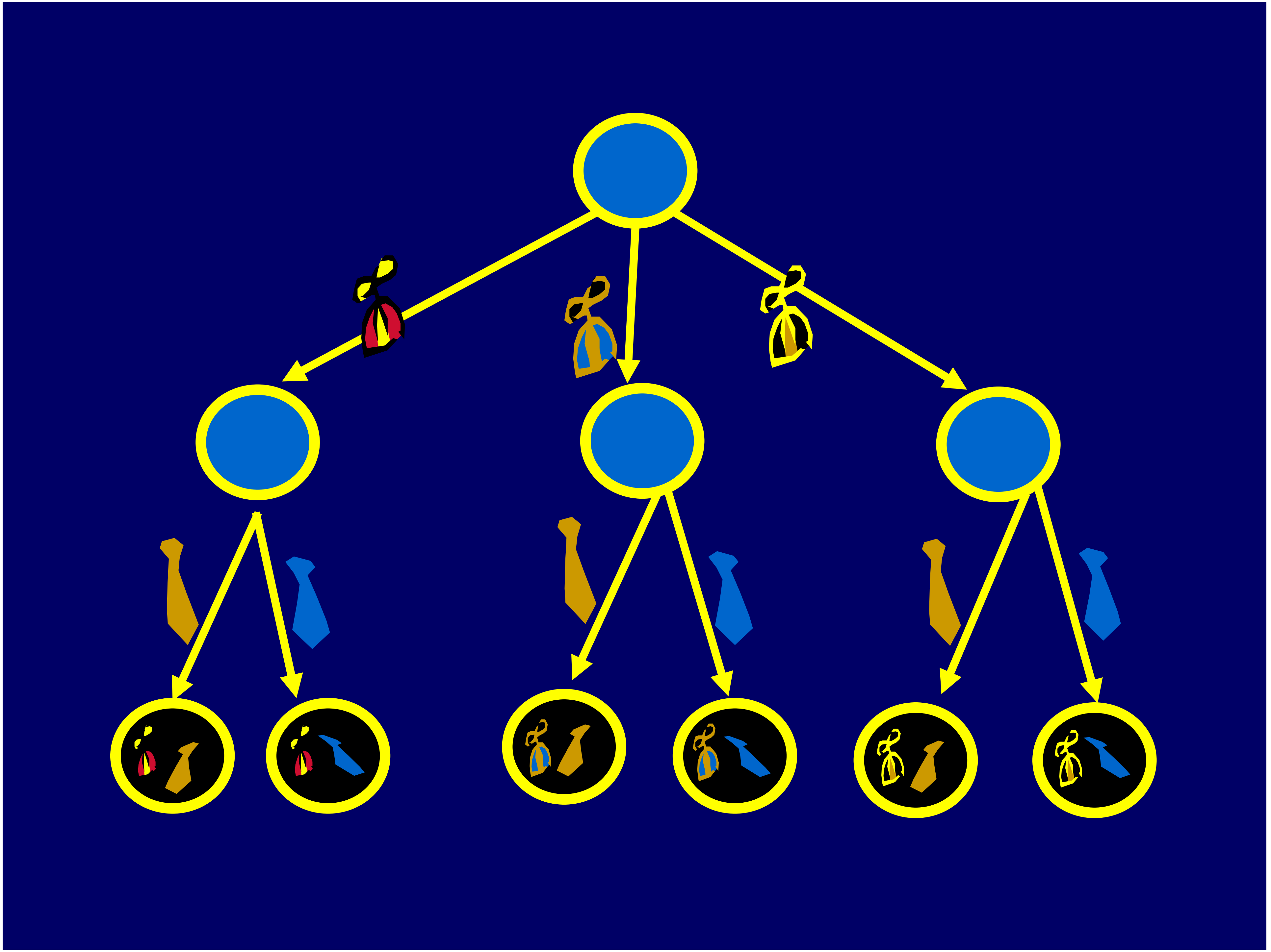
If G consists of root r with sub-trees $T_1, T_2, \ldots, T_k$

$Viz(G) =$

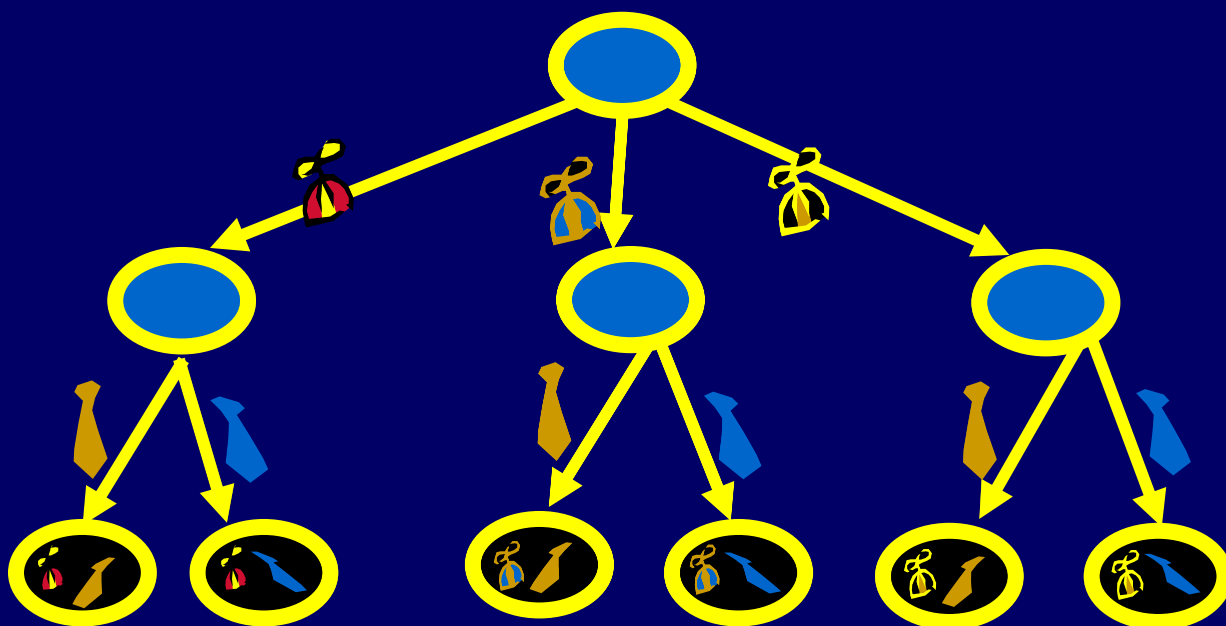



This visualization will be crucial to understanding properties of trees.


I own 3 beanies and 2 ties. How many beanie/tie combos might I wear to the ball tonight?



Choice Tree

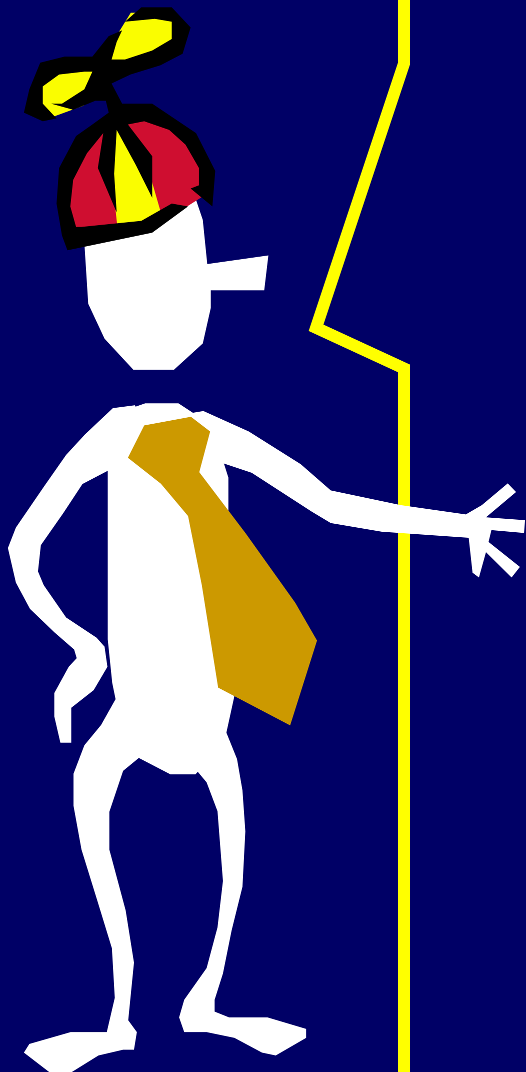


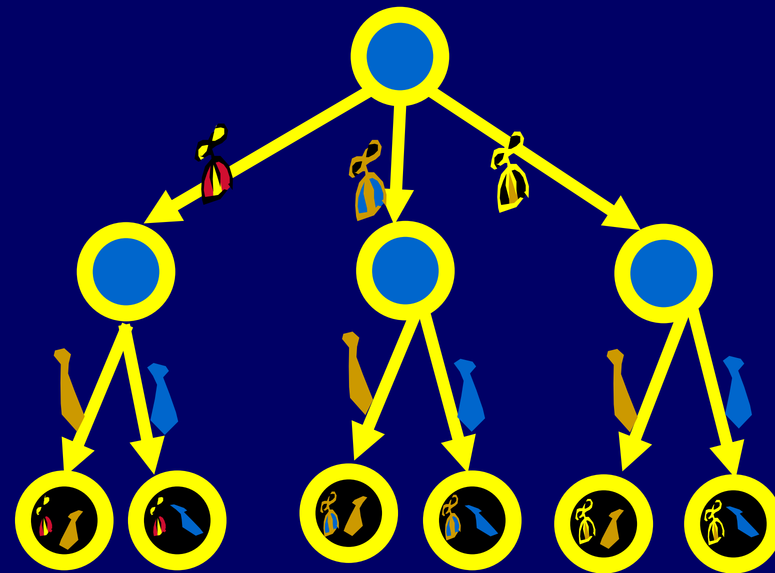
A <u>choice tree</u> is a tree with an object called a "choice" associated with each edge and a label called an "outcome" on each leaf.

Definition: Labeled Trees

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

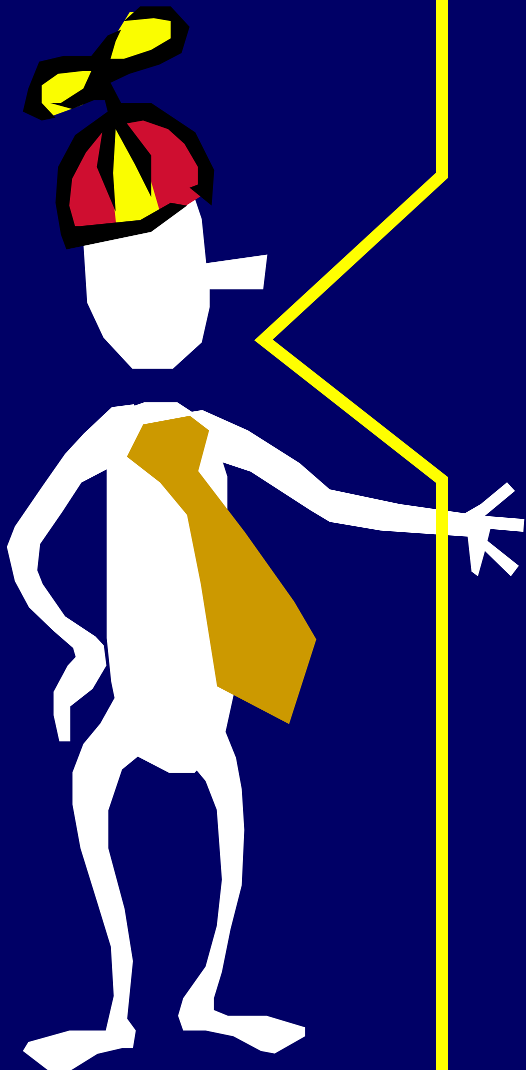
A tree "node labeled by S" is
a tree T = (V,E) with
an associated function Node-Label: $V \rightarrow S$





can be very illuminating.

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



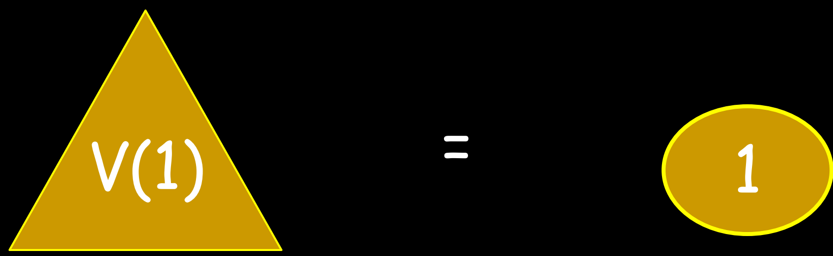
$T(1)=1;\ T(n)= 4T(n/2) + n$

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

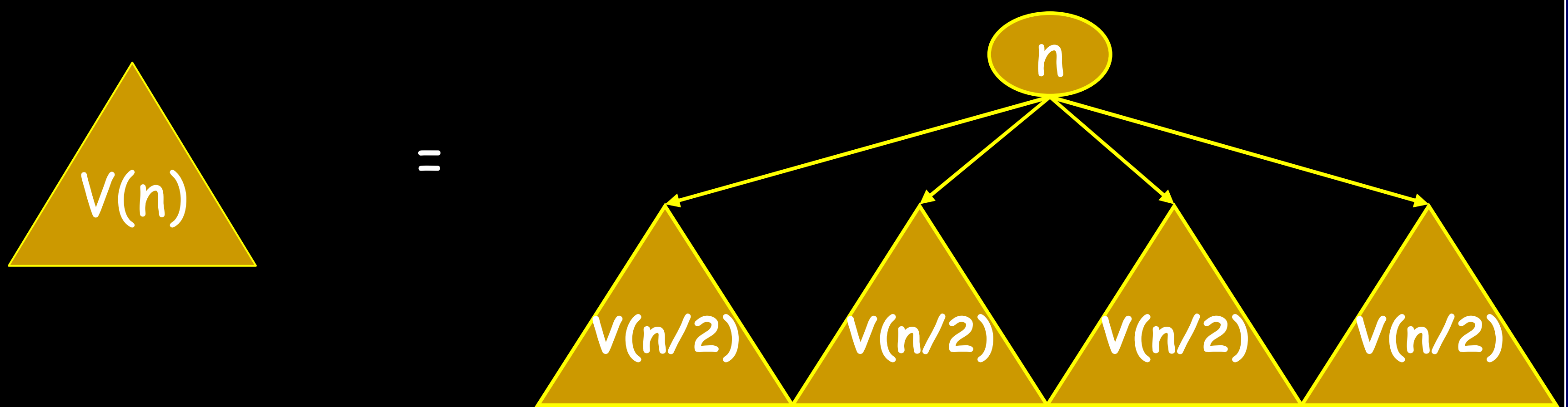
This tree $V(n)$ will give us an incisive picture of the recurrence relation $T(n)$.

Inductive Definition Of Labeled Tree V(n)

$T(1) = 1$



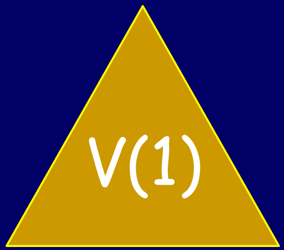
$T(n) = n + 4\ T(n/2)$



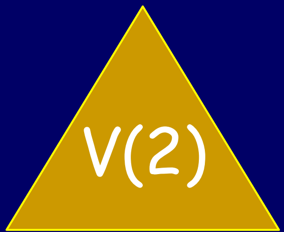
Inductive Definition Of Labeled Tree V(n)

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

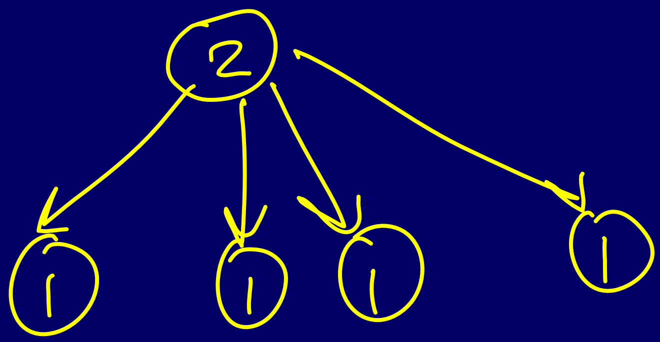
$T(1) \quad = \quad 1$



=



=

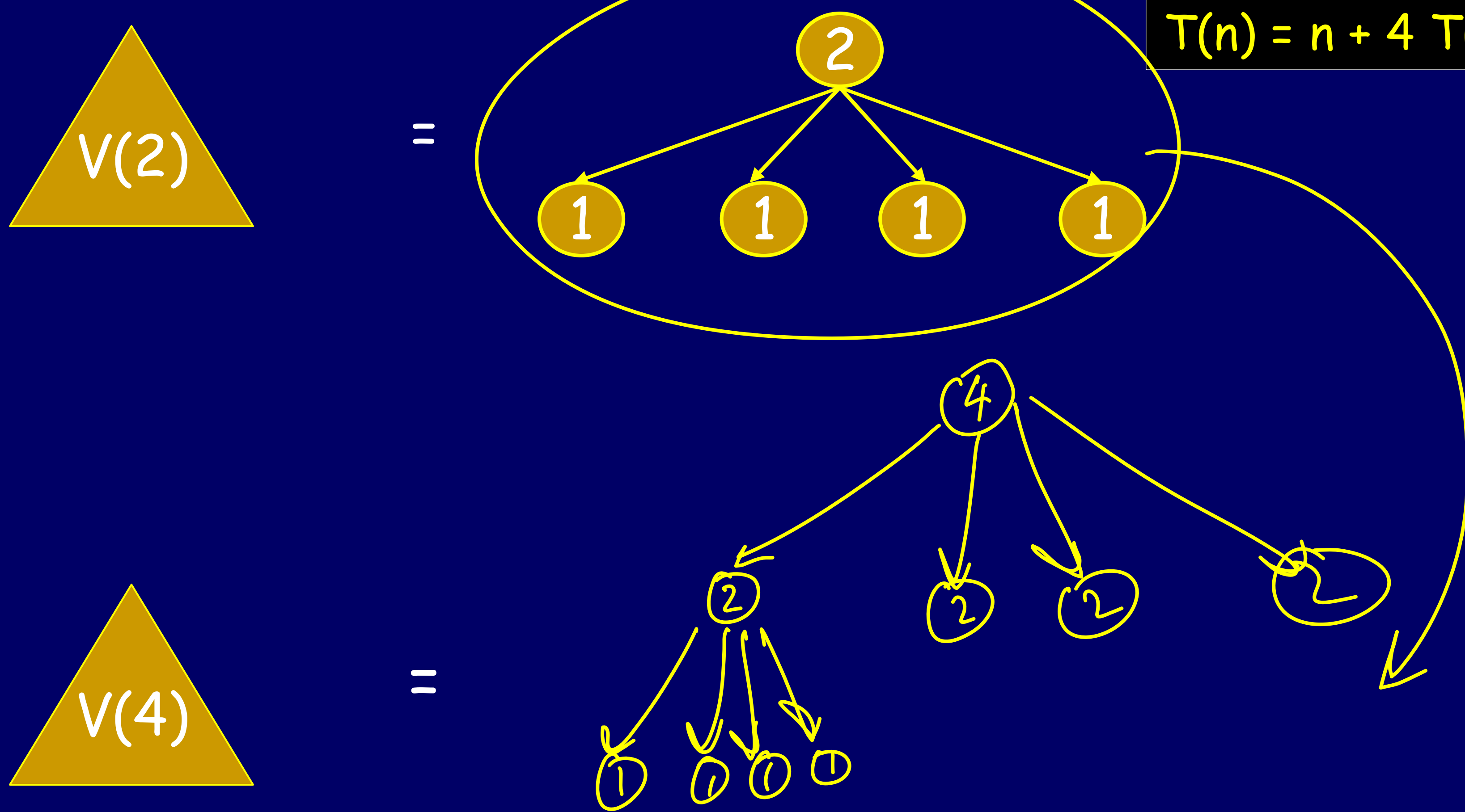


Inductive Definition Of Labeled Tree V(n)

$T(2) = 6$

$T(1) = 1$

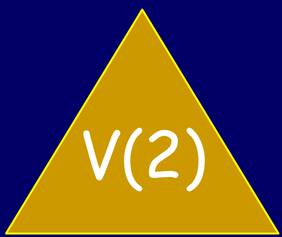
$T(n) = n + 4\,T(n/2)$



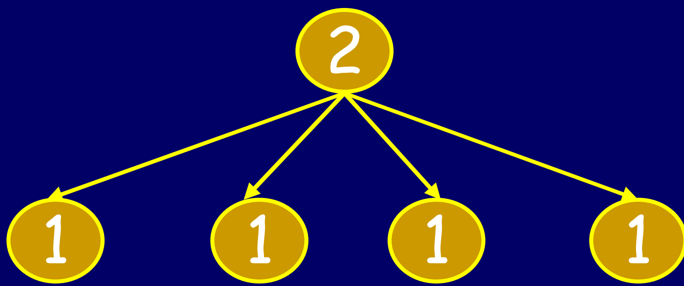
Inductive Definition Of Labeled Tree V(n)

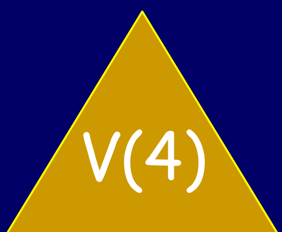
T(2) = 6

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

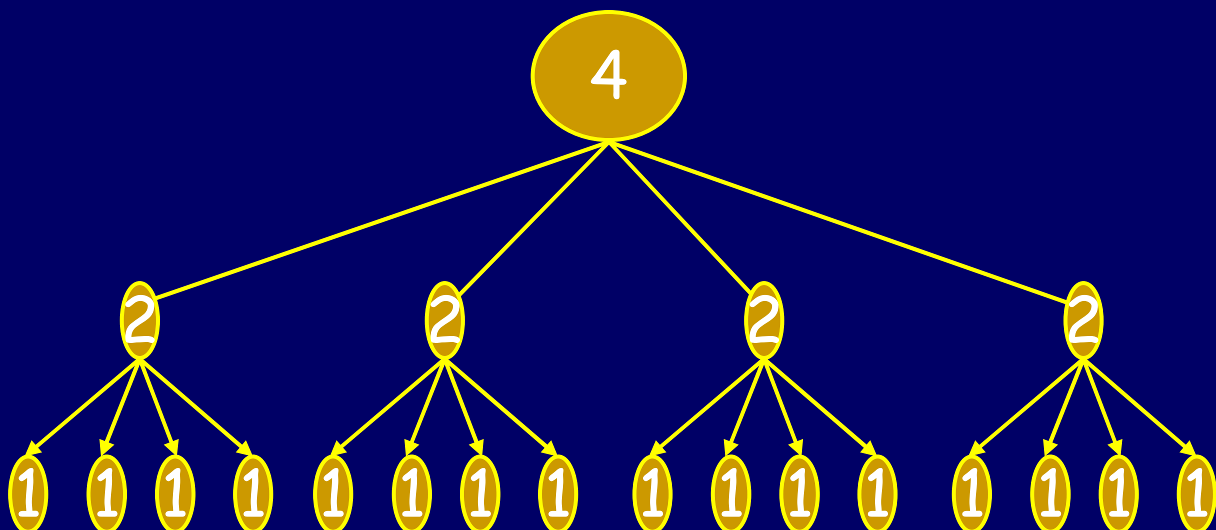


=





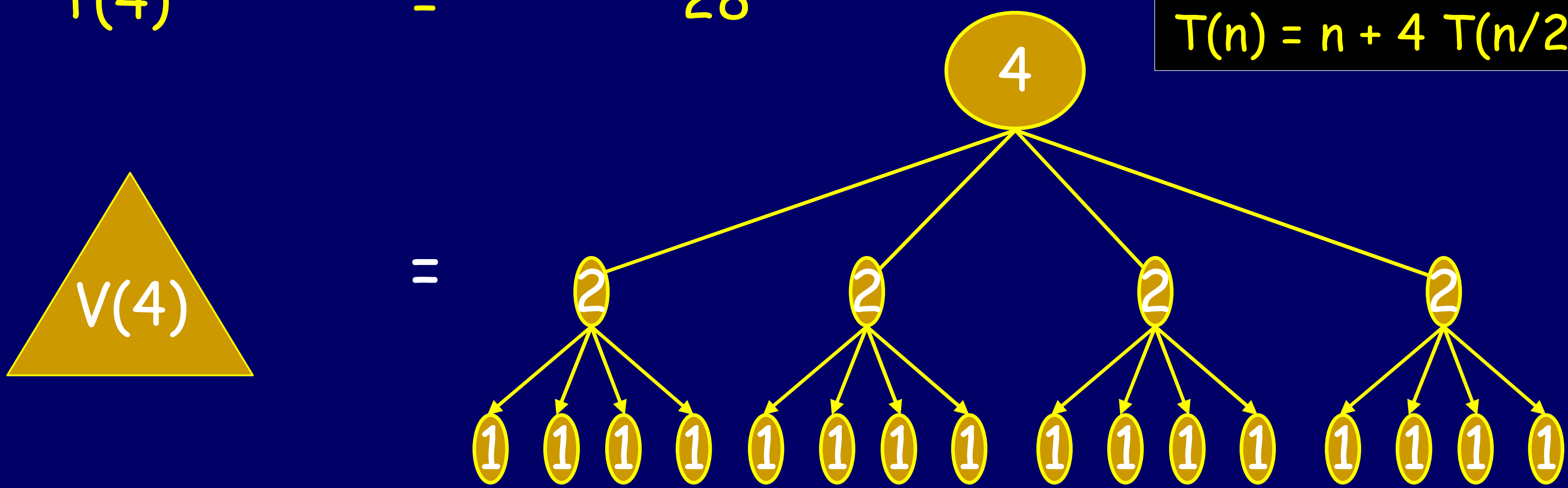
=



Inductive Definition Of Labeled Tree V(n)

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

T(4) = 28



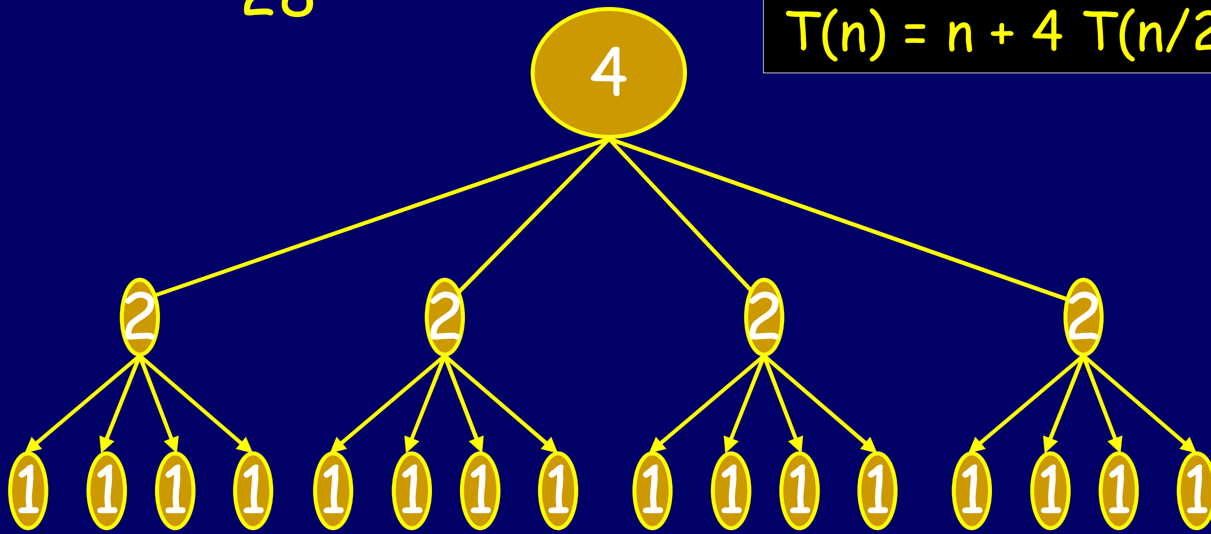
Inductive Definition Of Labeled Tree V(n)

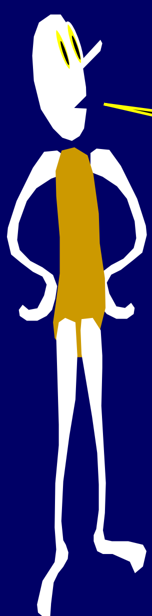
$T(1) = 1$

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

$T(4) = 28$

$V(4) =$

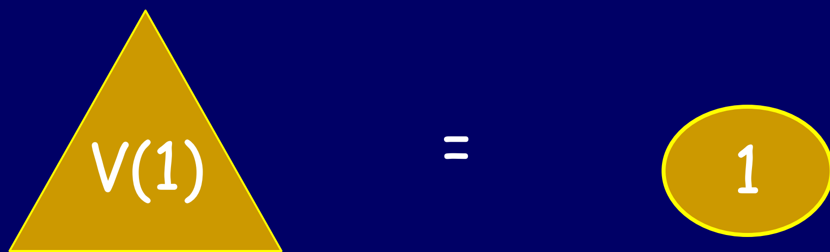




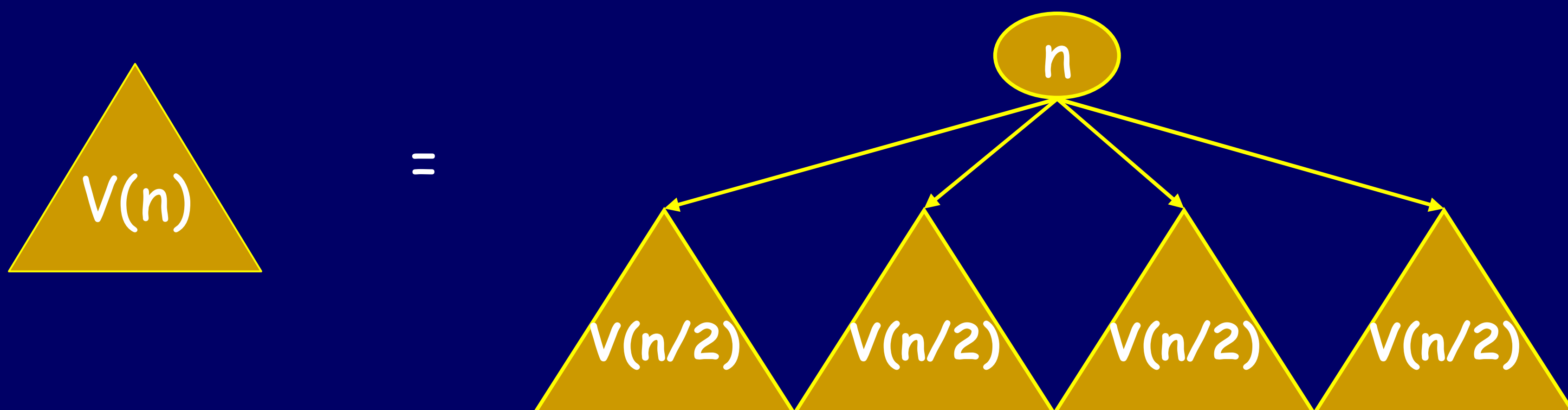
When we sum the node labels on V(n), we get back T(n)

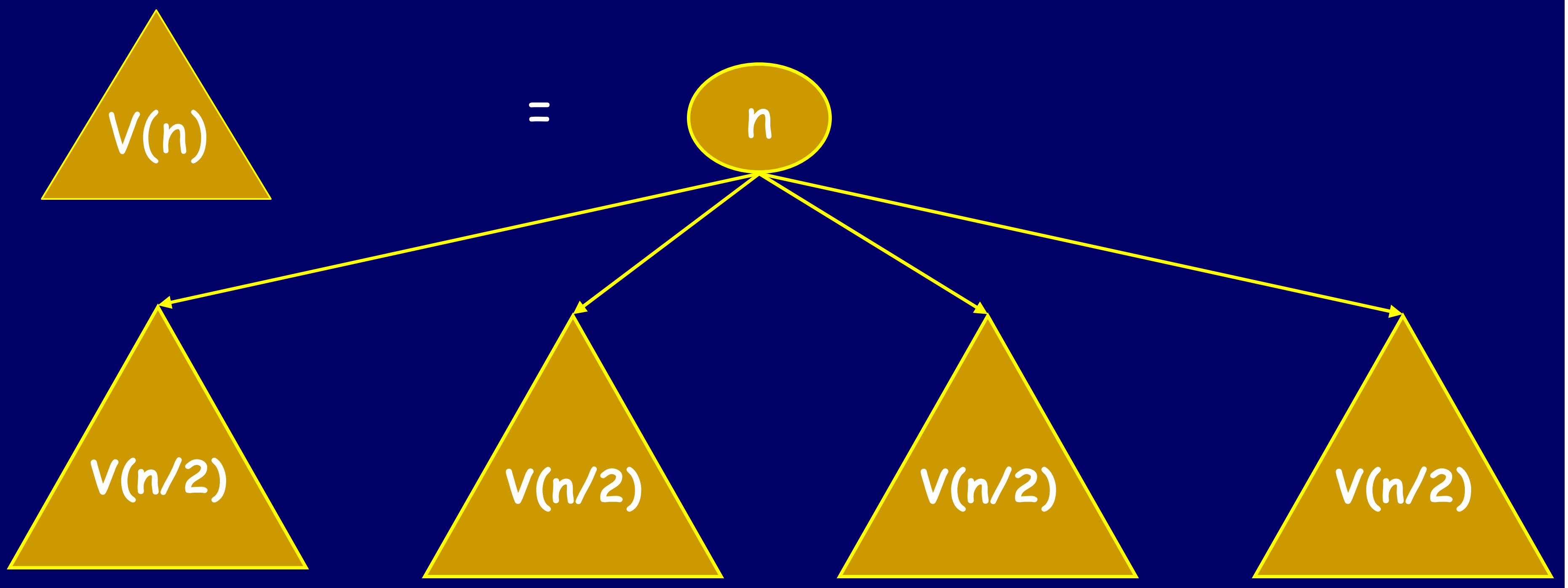
Can prove Invariant: $T(n) = $ (sum of labels in $V(n)$)

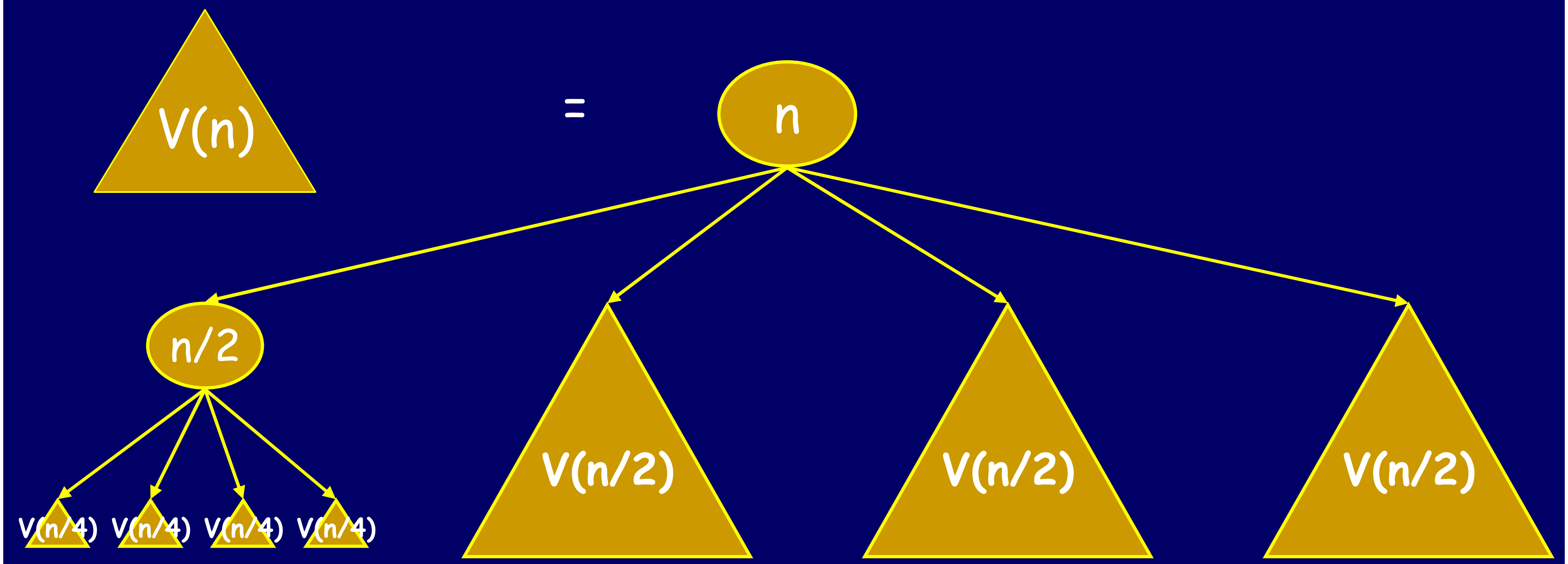
$$T(1) = 1$$

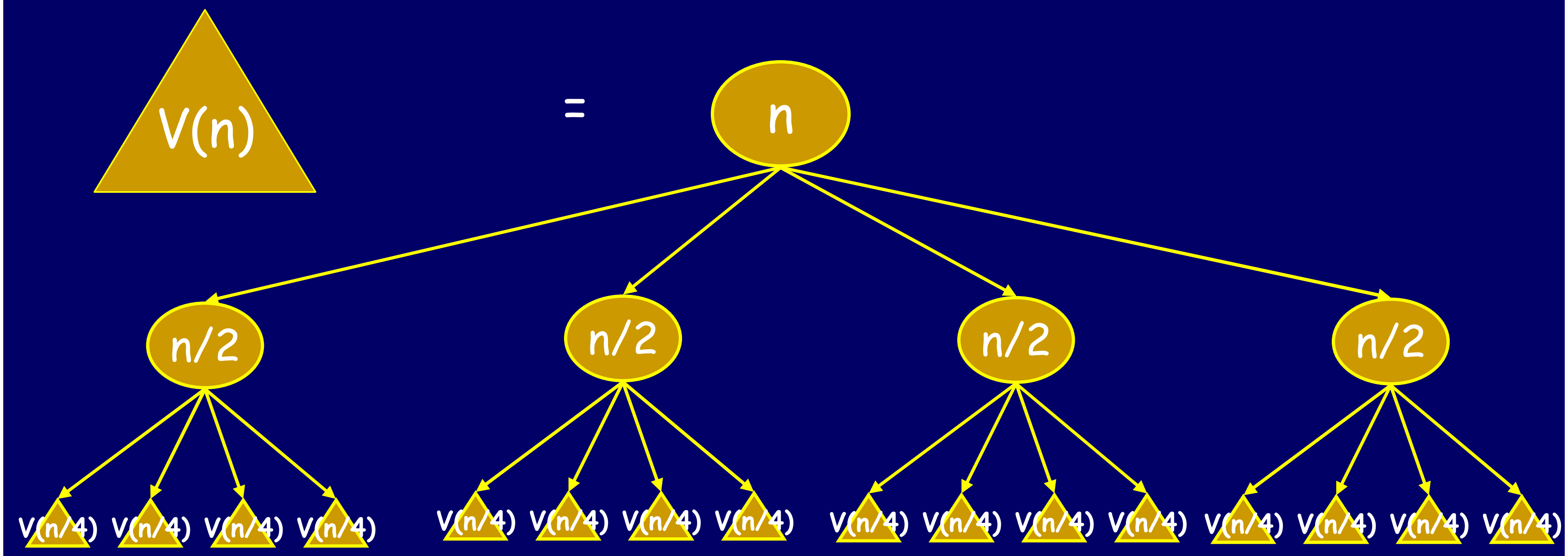


$$T(n) = n + 4\,T(n/2)$$

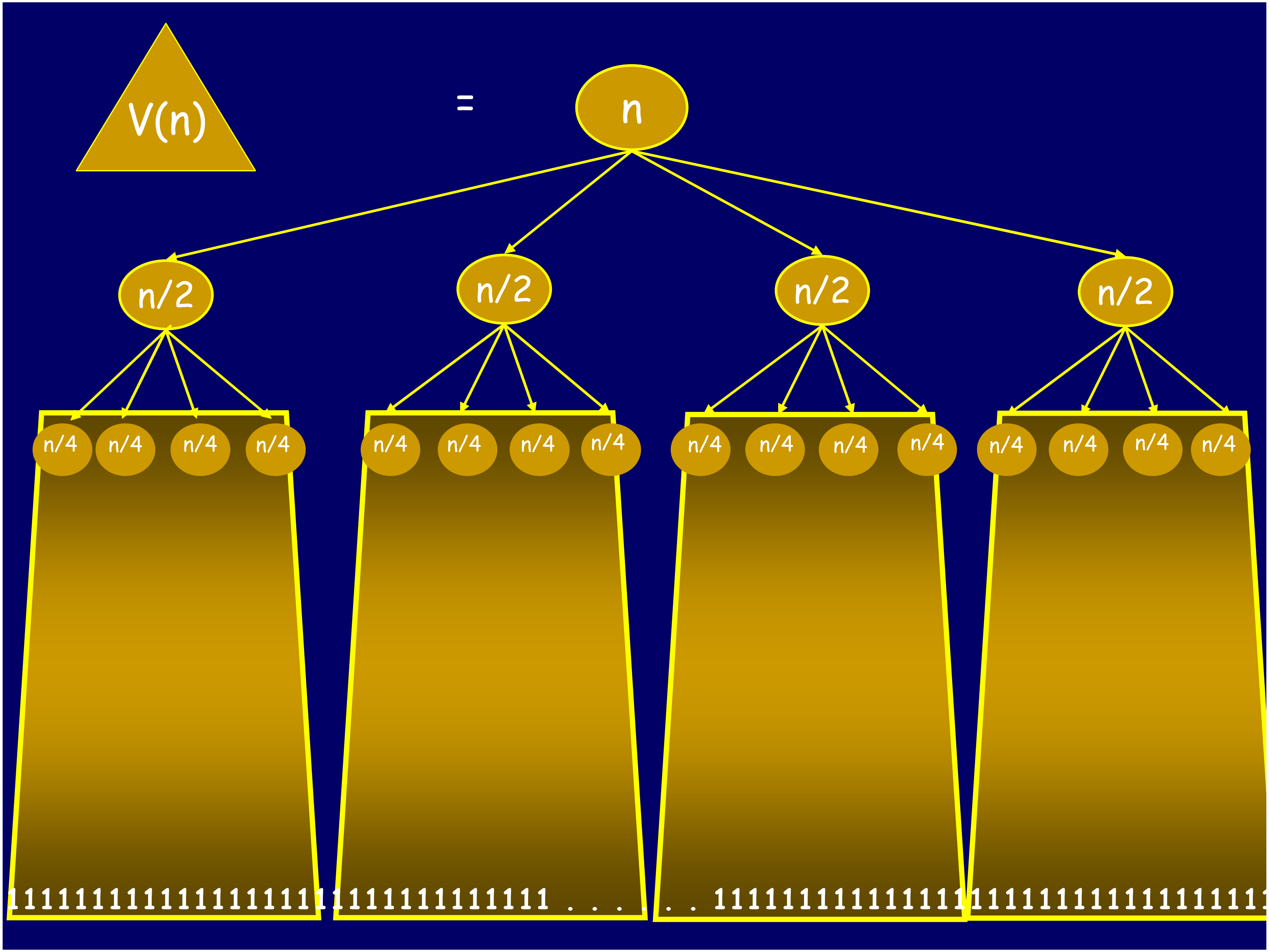



V(n)
=
n
V(n/2)
V(n/2)
V(n/2)
V(n/2)

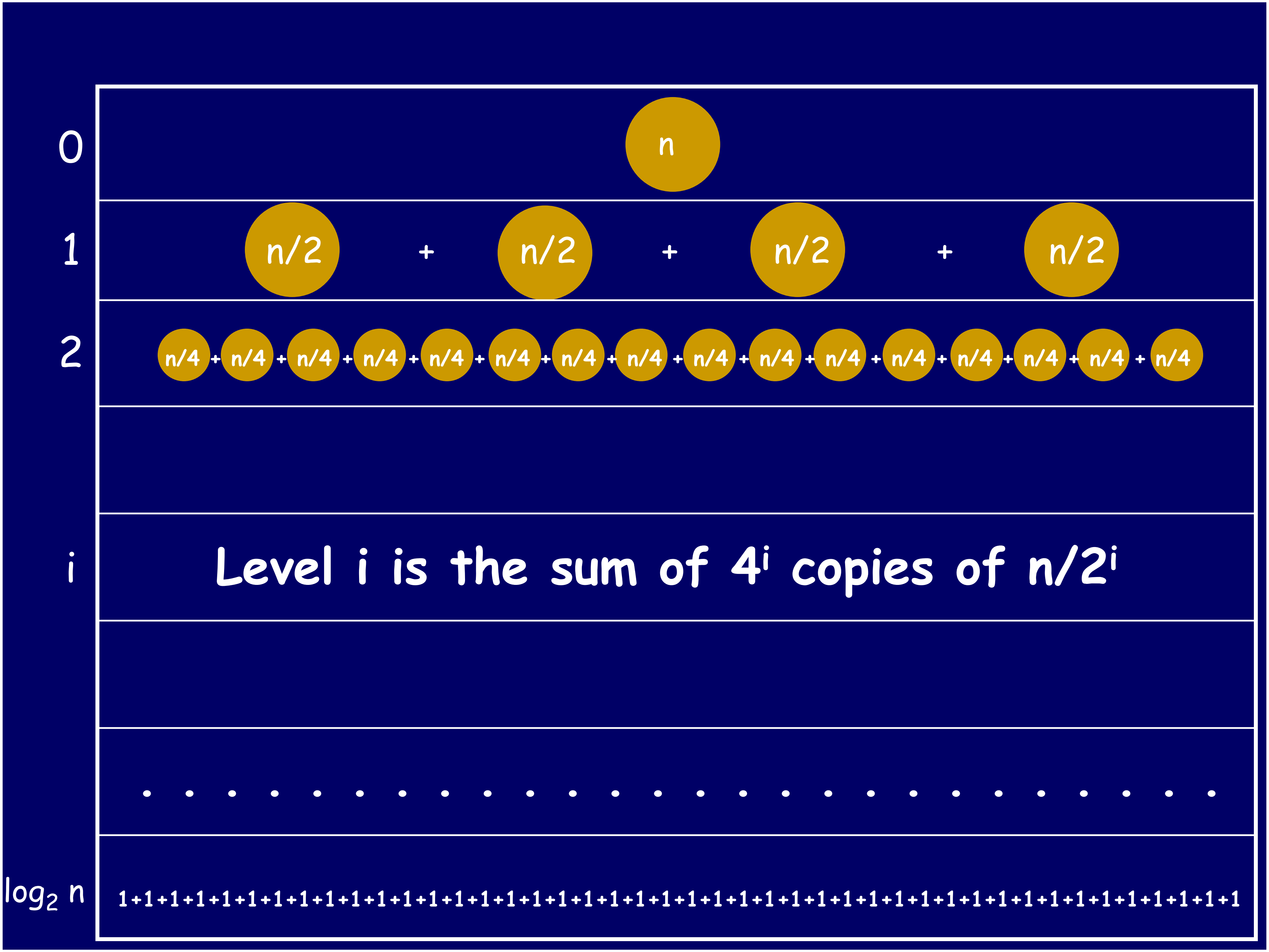

V(n)
=
n
n/2
V(n/2)
V(n/2)
V(n/2)
V(n/4) V(n/4) V(n/4) V(n/4)

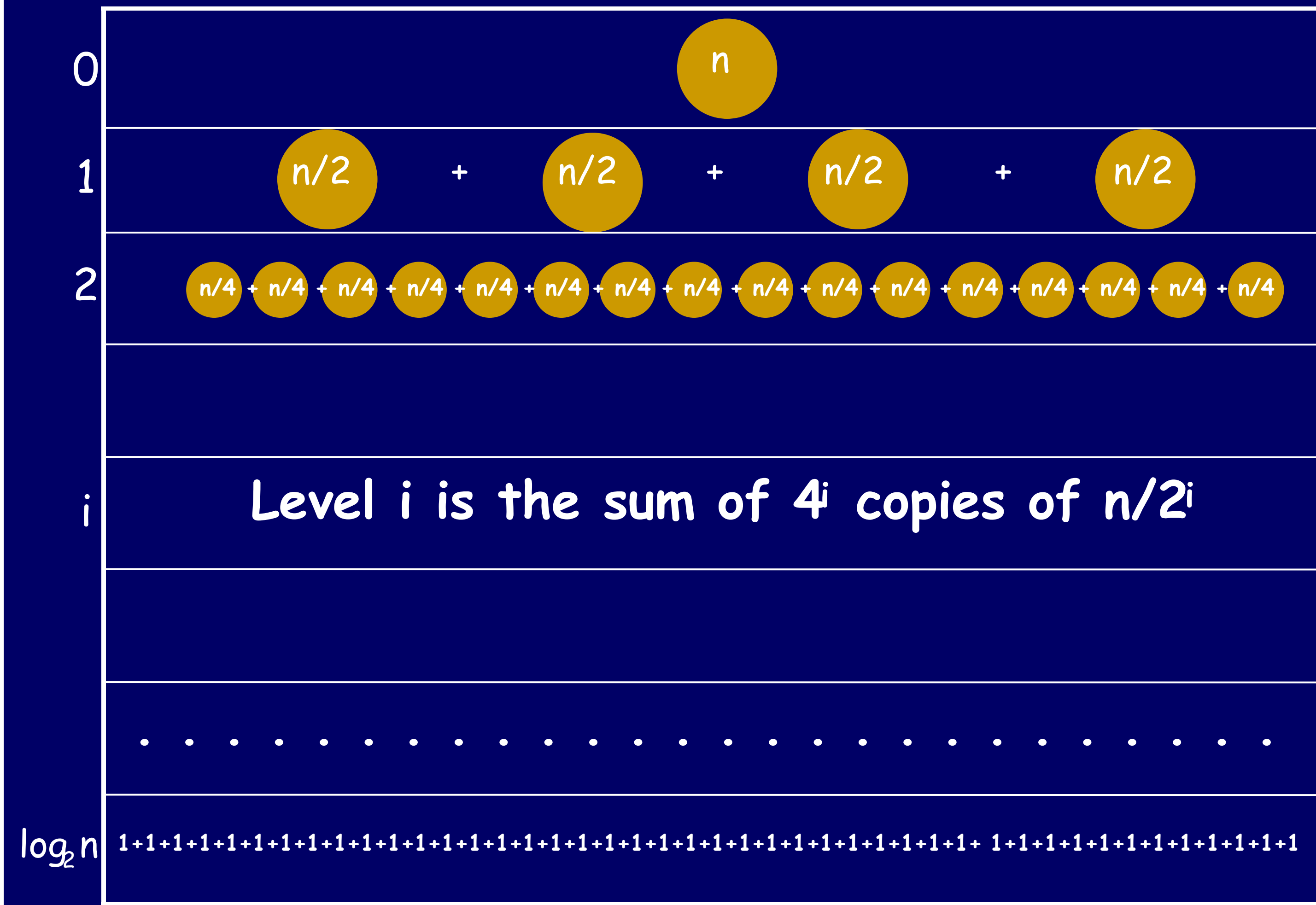


V(n)
=
n
n/2
n/2
n/2
n/2
V(n/4) V(n/4) V(n/4) V(n/4)
V(n/4) V(n/4) V(n/4) V(n/4)
V(n/4) V(n/4) V(n/4) V(n/4)
V(n/4) V(n/4) V(n/4) V(n/4)



V(n)
=
n
n/2
n/2
n/2
n/2
n/4 n/4 n/4 n/4
n/4 n/4 n/4 n/4
n/4 n/4 n/4 n/4
n/4 n/4 n/4 n/4
111111111111111111111111111111 1111111111111111111111111111111


0
n
1
n/2 + n/2 + n/2 + n/2
2
n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4
i
Level i is the sum of 4ⁱ copies of n/2ⁱ
log2 n
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+1+1+1+1+1+1+1+1+1+1+1+1



$= 1n$

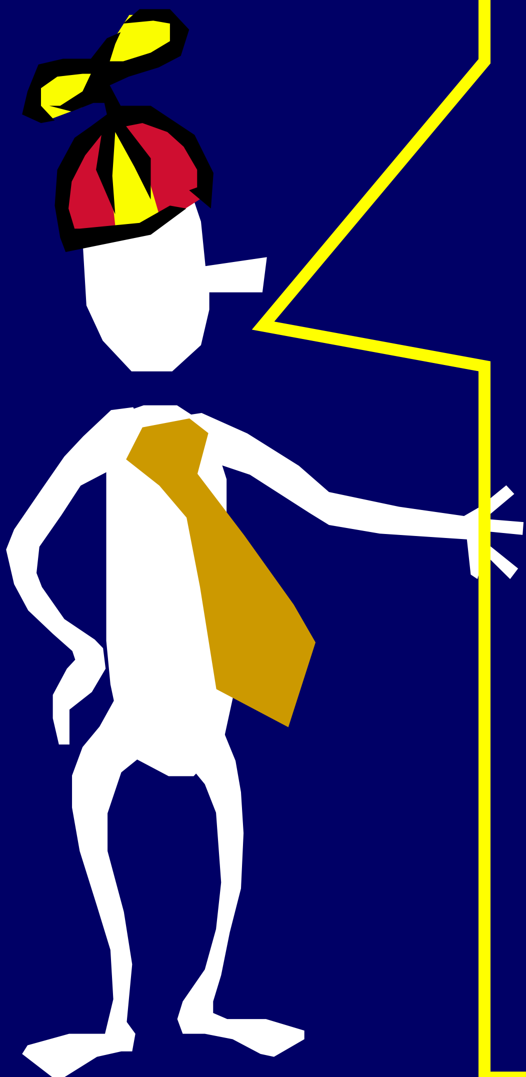
$= 2n$

$= 4n$

$= 2^i n$

$= (n)n$

$n(1+2+4+8+ \ldots +n) =$ $n(2n-1) =$ $2n^2-n$



And the technique is very general.

Instead of

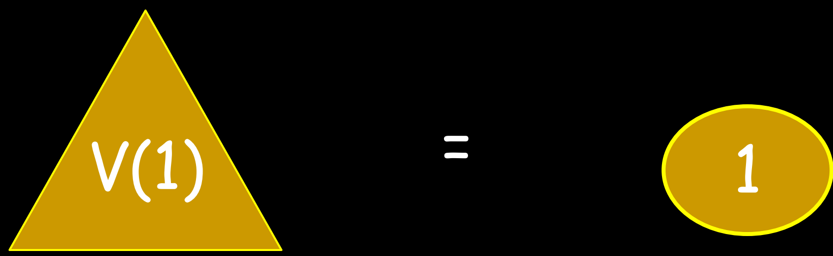
$T(1)=1;\ T(n)= 4T(n/2) + n$

We could illuminate

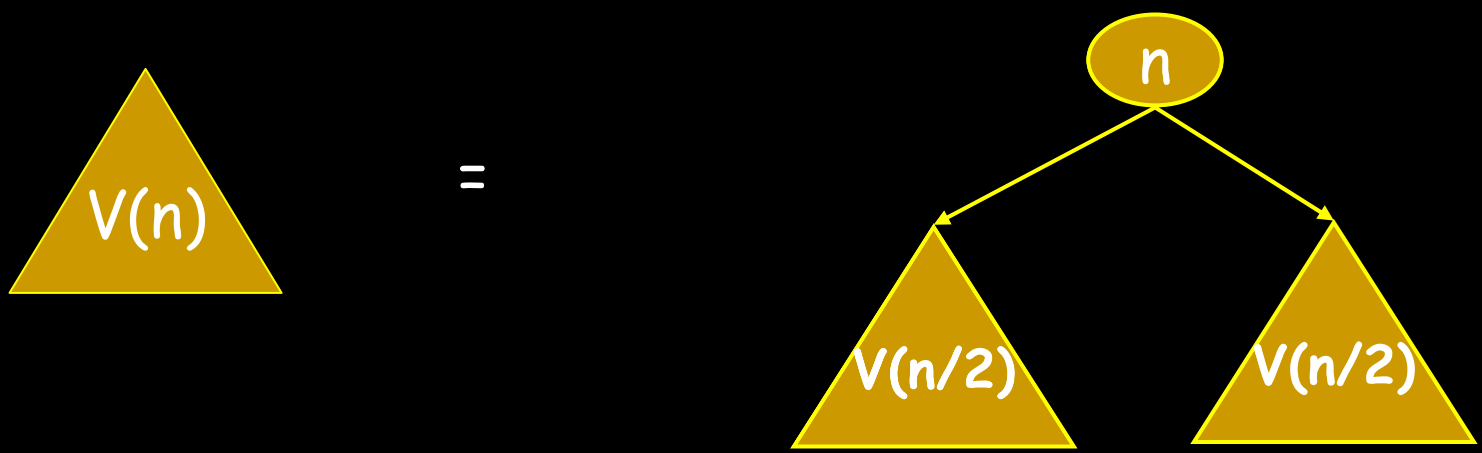
$T(1)=1;\ T(n) = 2T(n/2) + n$

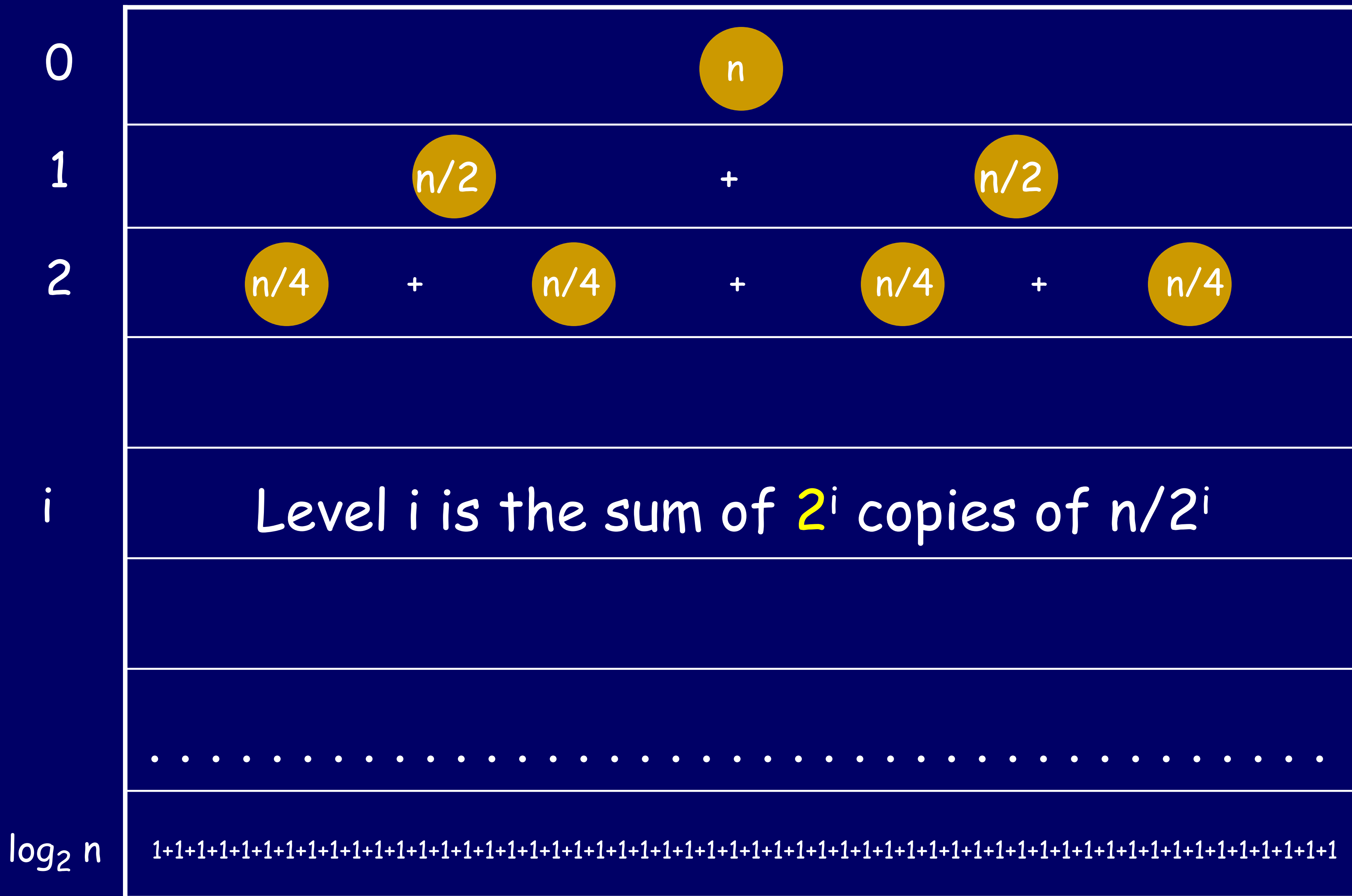
New Recurrence $T(n) = 2T(n/2) + n$

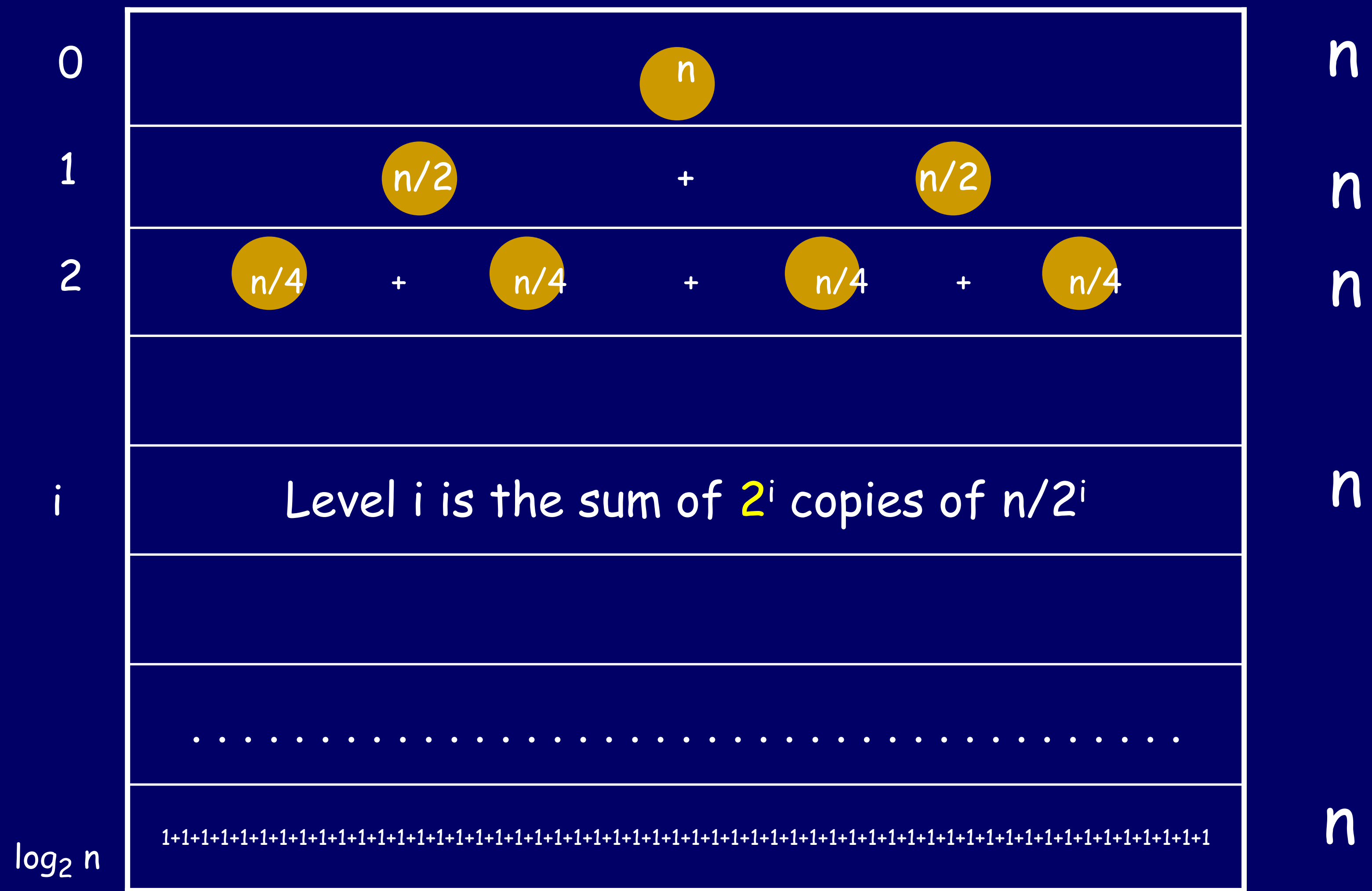
$T(1) \quad = \quad 1$



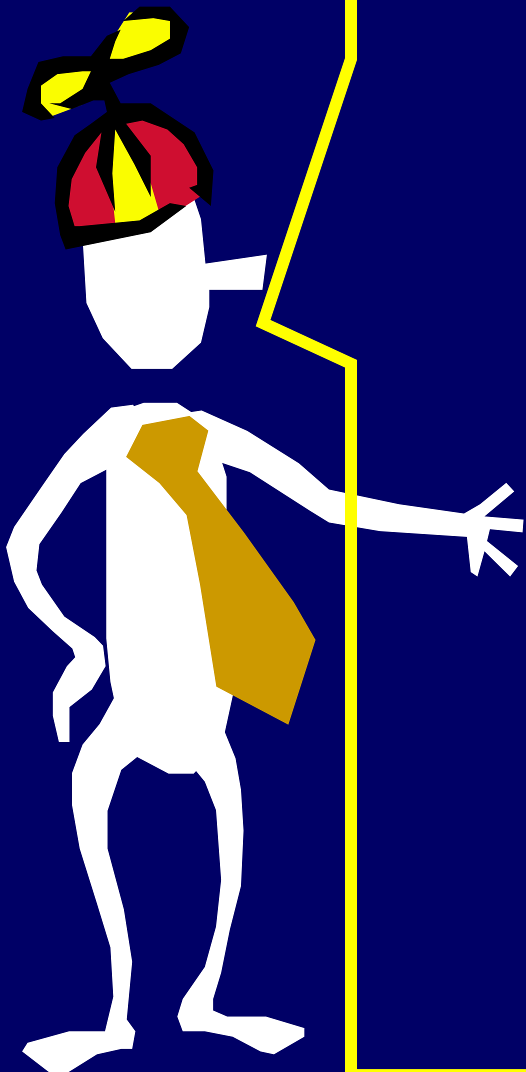
$T(n) \quad = \quad n + 2\,T(n/2)$




0
n
1
n/2
+
n/2
2
n/4
+
n/4
+
n/4
+
n/4
i
Level i is the sum of 2^i copies of n/2^i
. . . .
log2 n
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+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



$$n(1+1+1+\ldots+1) = \qquad n\,(\log_2 n + 1)$$

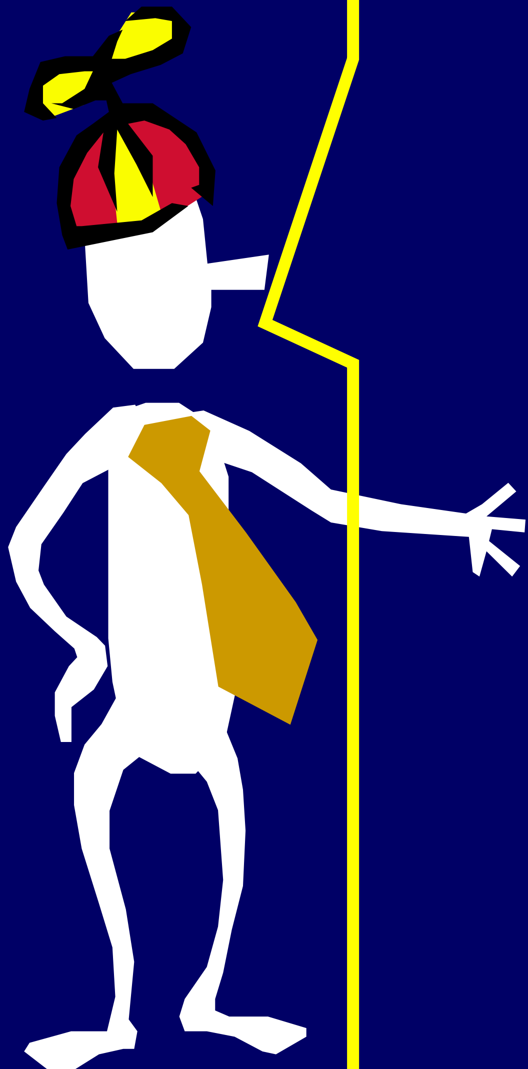


$$T(1)=1;\ T(n) = 2T(n/2) + n$$

has closed form

$$T(n) = n\,(\log_2(n) + 1)$$

where n is a power of 2



$$T(n) = A\,T(n/B) + n$$

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: $\Sigma = \{a,b\}$

Start word: a

Production Rules:

Sub(a) = ab

Sub(b) = a

$\text{NEXT}(w_1\ w_2\ \ldots\ w_n) = \text{Sub}(w_1)\ \text{Sub}(w_2)\ \ldots\ \text{Sub}(w_n)$

a
ab
aba
abaab
abaababa
abaababaabaab

How long are the strings as a function of time?

FIBONACCI(n) at time n

Aristid Lindenmayer (1925-1989)

1968 Invents L-systems in Theoretical Botany

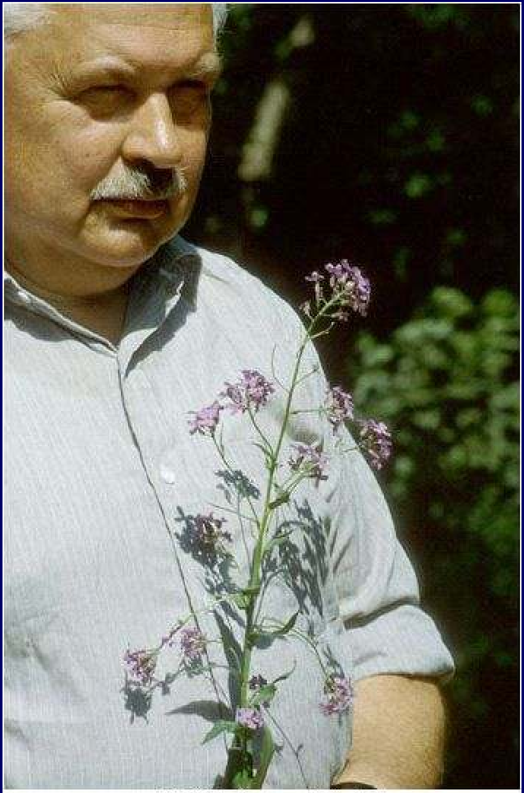
Time 1: a

Time 2: ab

Time 3: aba

Time 4: abaab

Time 5: abaababa



Aristid Lindenmayer 1925–1989

The Koch Game

Helge von Koch

Alphabet: $\Sigma = \{ F, +, - \}$

Start word: F

Production Rules:

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

Sub(+) = +

Sub(-) = -

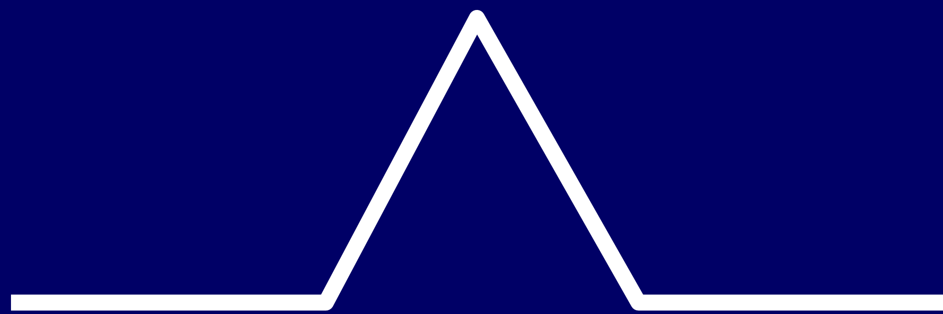
NEXT$(w_1\ w_2\ \ldots\ w_n)$ = Sub(w_1) Sub(w_2) … Sub(w_n)

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

The Koch Game

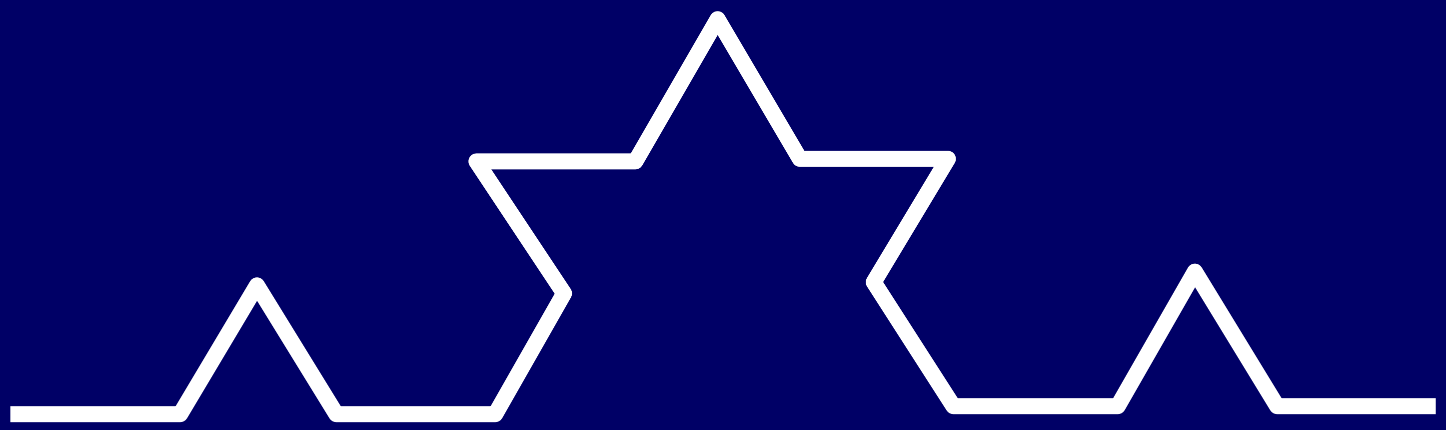


F+F--F+F

Visual representation:

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

The Koch Game

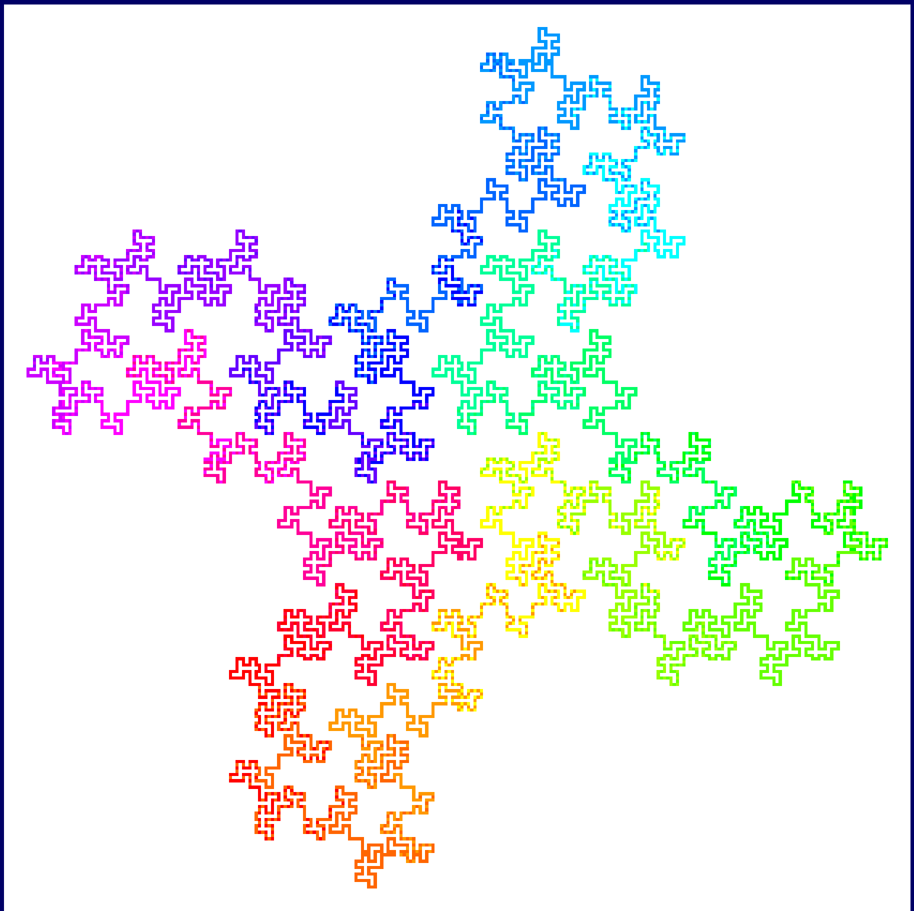


F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

Visual representation:

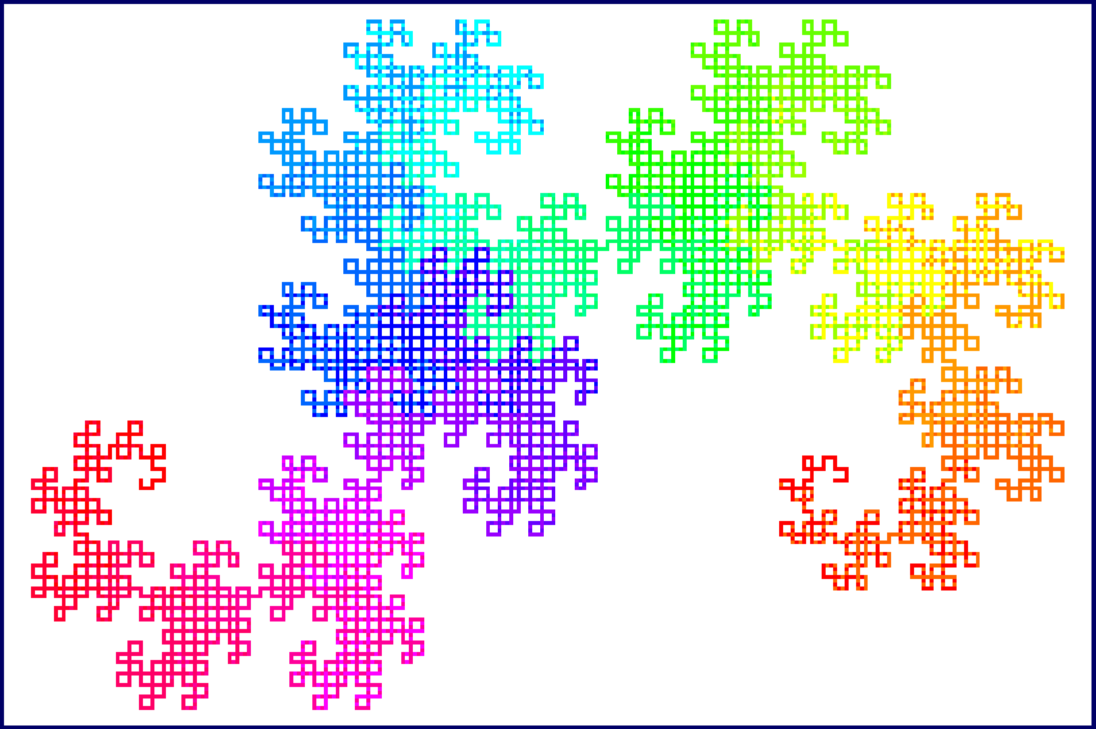
F	draw forward one unit
+	turn 60 degree left
-	turn 60 degrees right

Koch Curve



Dragon Game

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

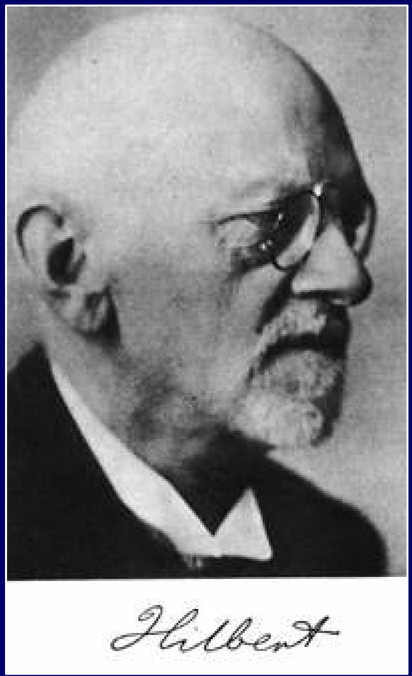


Dragon Curve

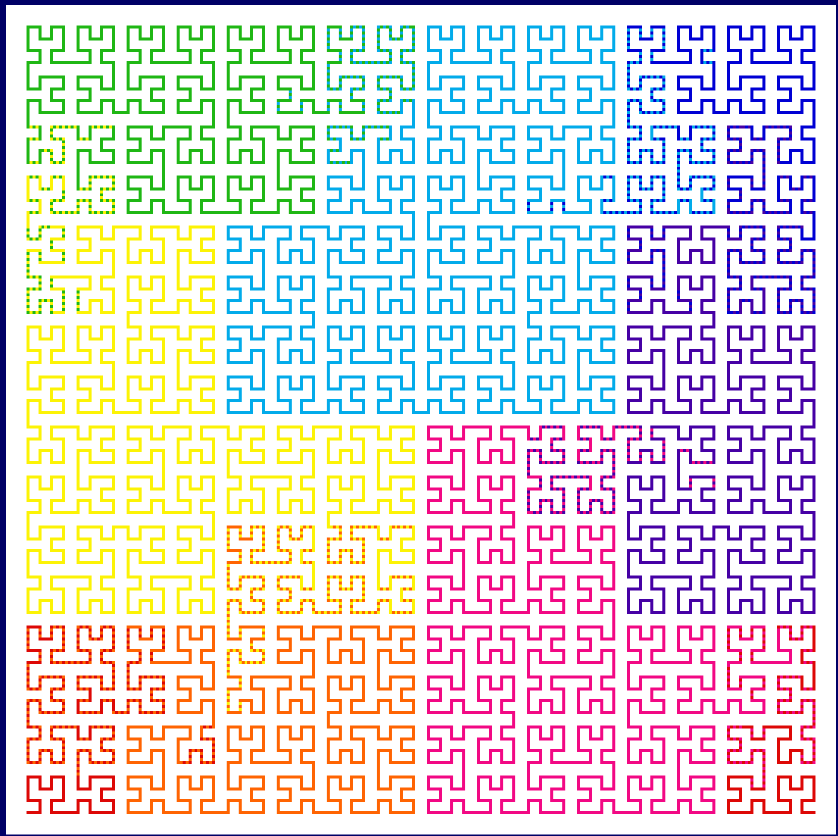
Hilbert Game

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

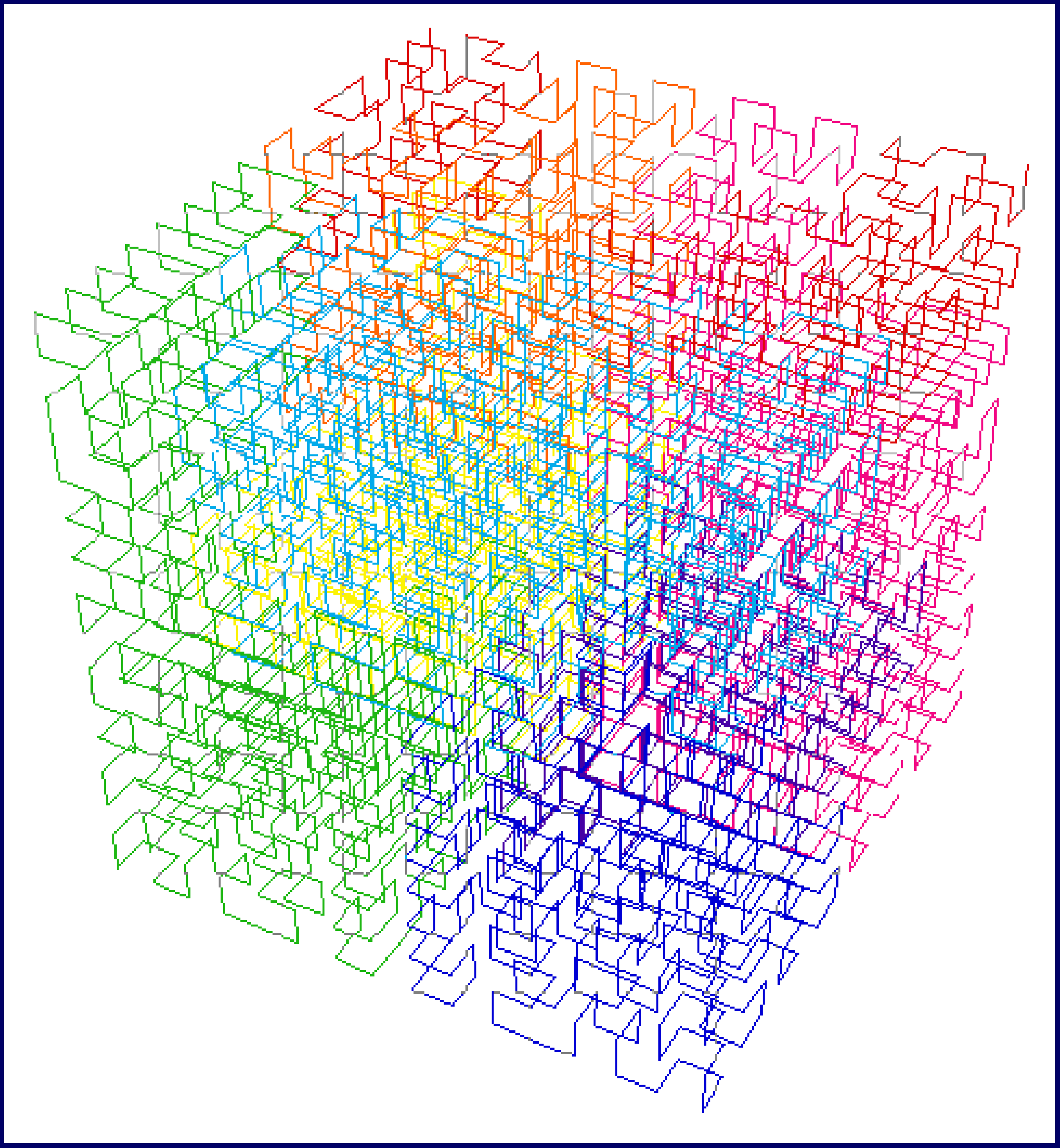
Make 90 degree turns instead of 60 degrees.



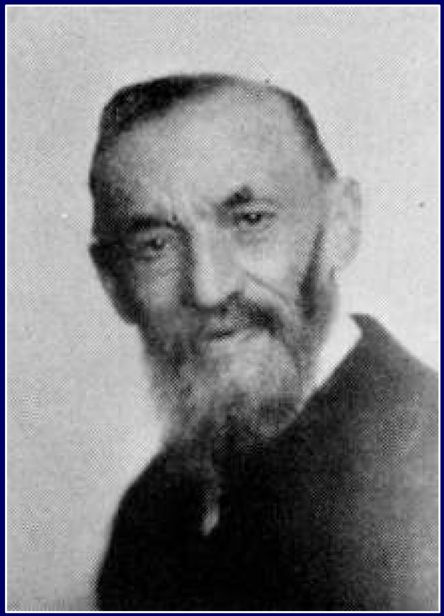
Hilbert Curve

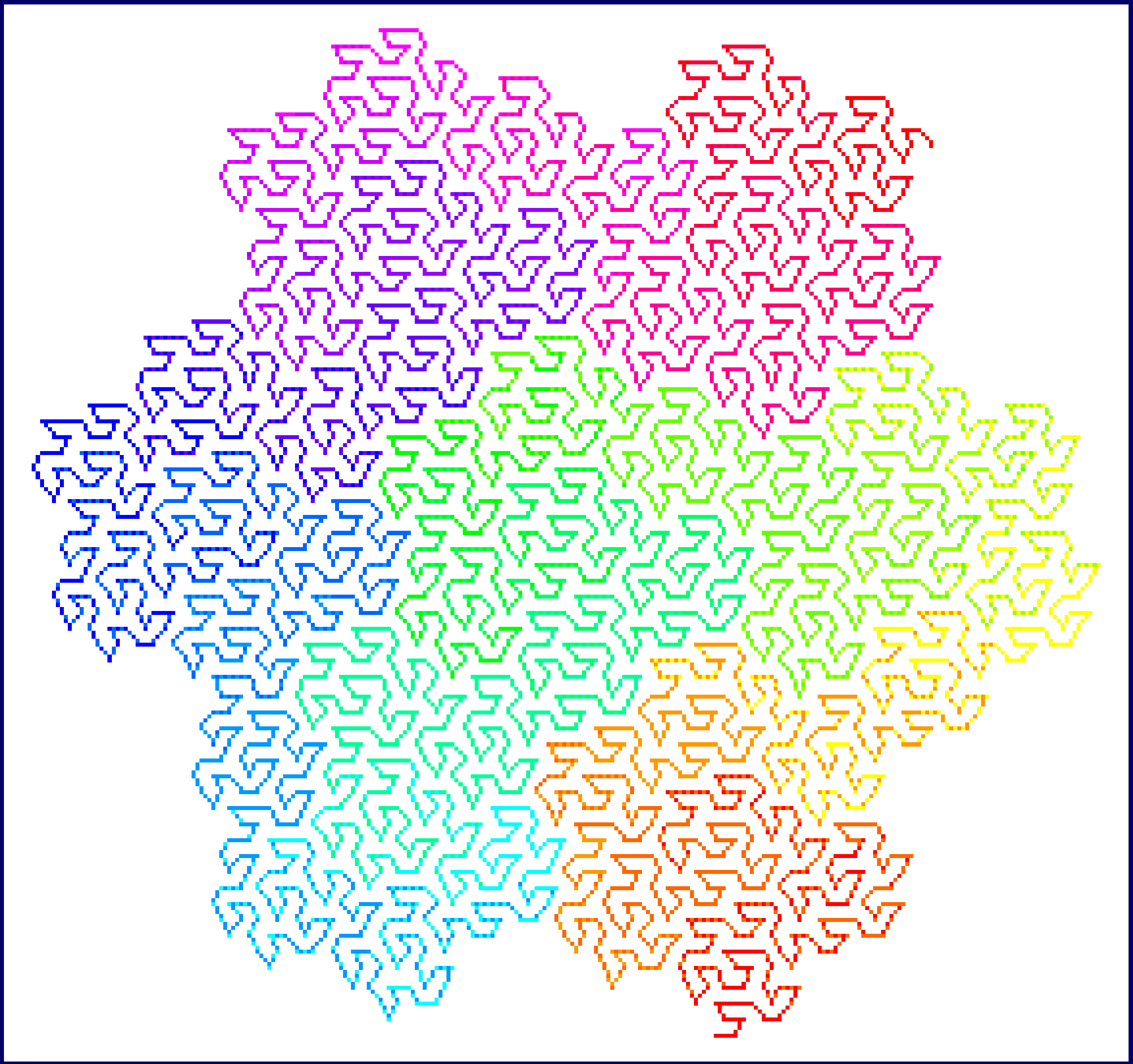


Hilbert's space filling curve

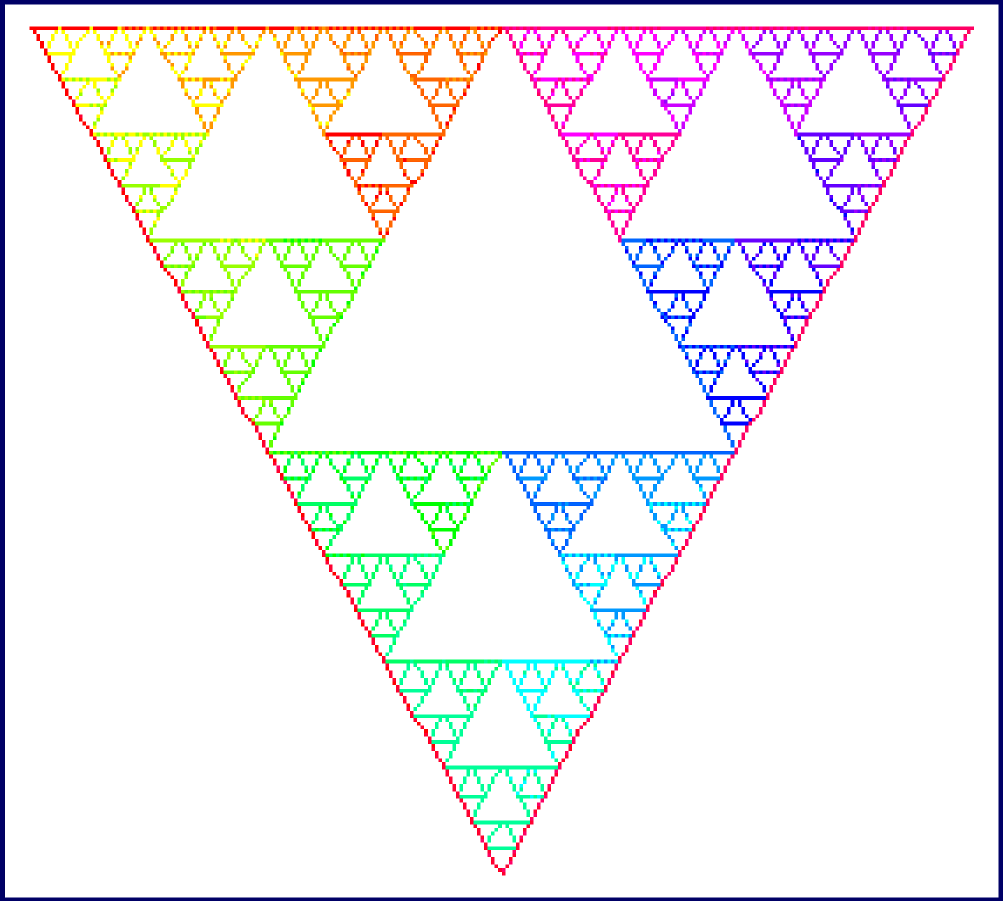


Peano's gossamer curve





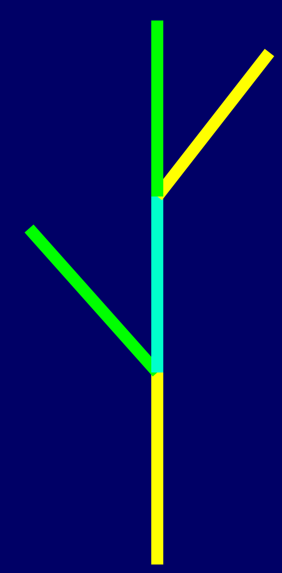
Sierpinski's triangle



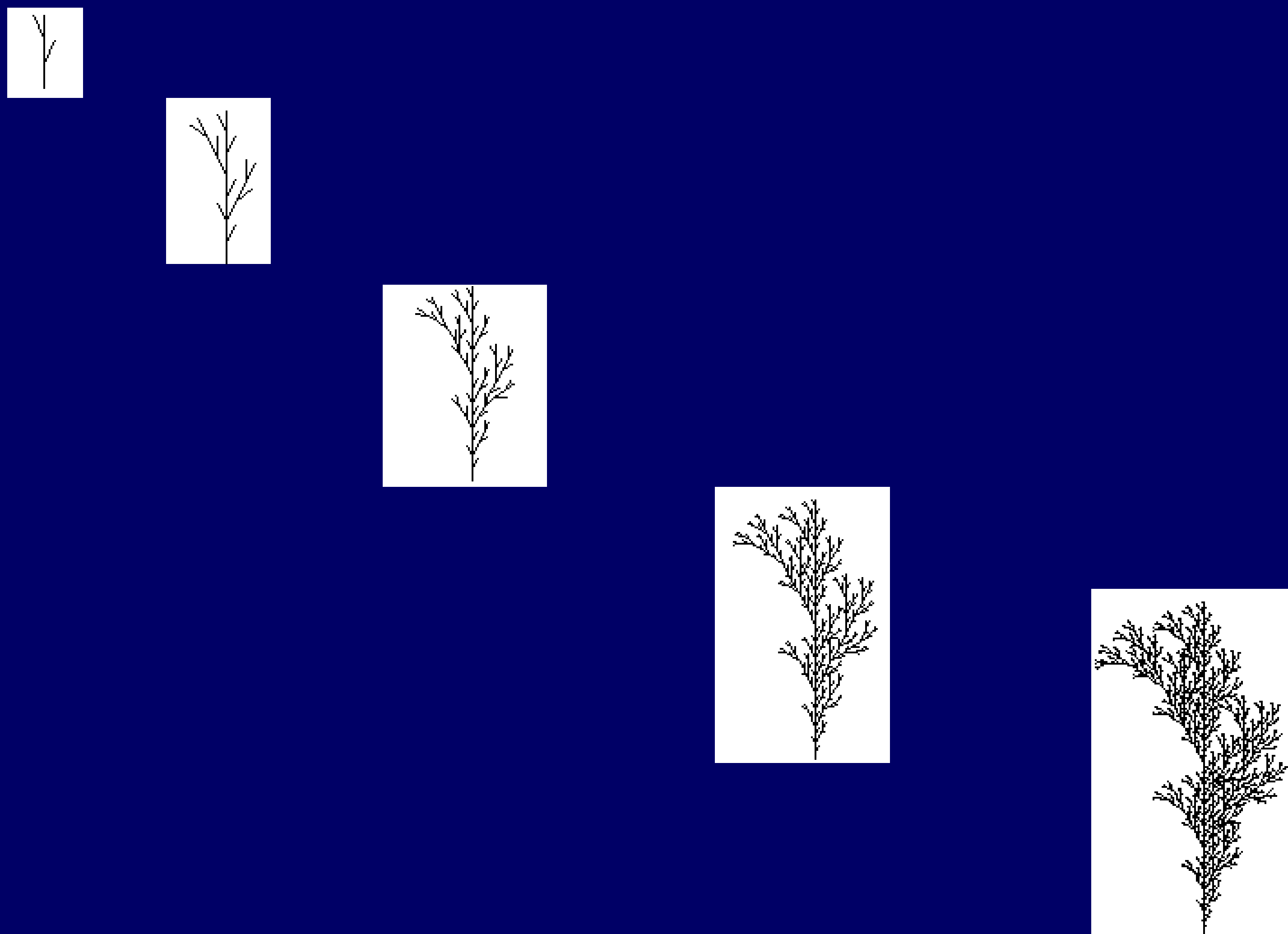
Lindenmayer 1968

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

Interpret the stuff inside brackets as a branch.



Lindenmayer 1968



Inductive Leaf

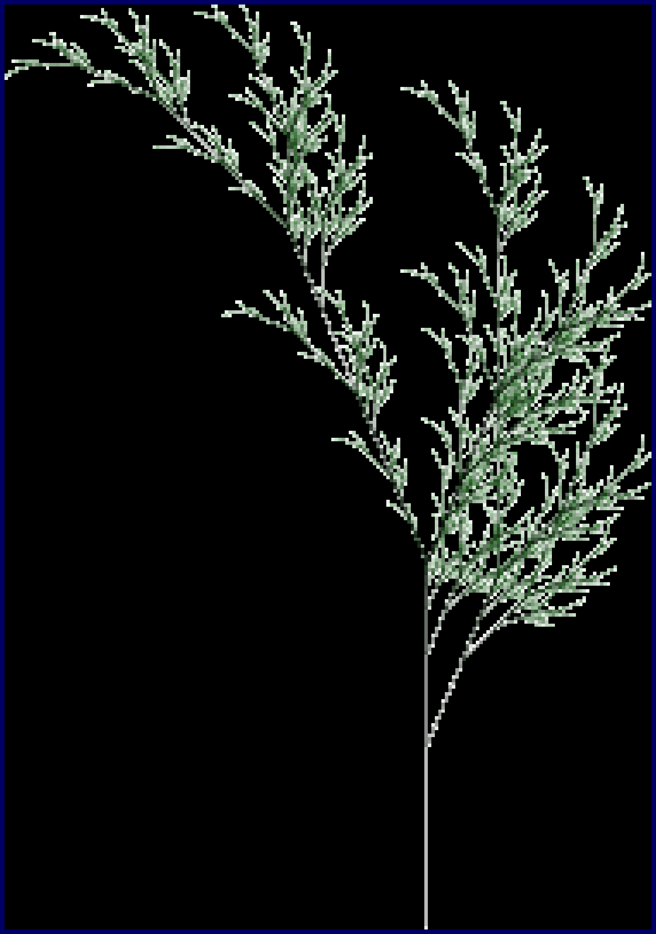
"The Algorithmic Beauty of Plants"

Start at X

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

Sub(F) = FF

Angle=22.5



©*The Algorithmic Beauty of Plants*, Przemyslaw Prusinkiewicz and Aristid Lindenmayer, Springer-Verlag 1990

Much more stuff at

http://www.cbc.yale.edu/courseware/swinglsystem.html

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