Recursive Definitions

And Applications to Counting

C(n,k)

- $C(n,k) = C(n-1,k-1) + C(n-1,k)$ (where $n,k \geq 1$)
  - Easy derivation: Let $|S|=n$ and $a \in S$.

    $C(n,k) =$ # k-sized subsets of S containing a

    + # k-sized subsets of S not containing a
- In fact, gives a recursive definition of $C(n,k)$
  - Base case (to define for $k \leq n$): $C(n,0) = C(n,n) = 1$ for all $n \in \mathbb{N}$
  - Or, to define it for all $(n,k) \in \mathbb{N} \times \mathbb{N}$

    Base case: $C(n,0)=1$, for all $n \in \mathbb{N}$, and $C(0,k)=0$ for all $k \in \mathbb{Z}^+$

n \ k	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
2	1	2	1	0	0	0	0
3	1	3	3	1	0	0	0
4	1	4	6	4	1	0	0
5	1	5	10	10	5	1	0
6	1	6	15	20	15	6	1

Tower of Hanoi

- Move entire stack of disks to another peg
  - Move one from the top of one stack to the top of another
  - A disk cannot be placed on top of a smaller disk
- How many moves needed?
- Optimal number not known when 4 pegs and over ≈30 disks!
- Optimal solution known for 3 pegs (and any number of disks)

Tower of Hanoi

- Recursive algorithm (optimal for 3 pegs)
  - Transfer(n,A,C):

    If n=1, move the single disk from peg A to peg C

    Else

      Transfer(n-1,A,B) (leaving the largest disk out of play)

      Move largest disk to peg C

      Transfer(n-1,B,C) (leaving the largest disk out of play)

Tower of Hanoi

- Recursive algorithm (optimal for 3 pegs)
  - Transfer(n,A,C):

    If n=1, move the single disk from peg A to peg C

    Else

    Transfer(n-1,A,B) (leaving the largest disk out of play)

    Move largest disk to peg C

    Transfer(n-1,B,C) (leaving the largest disk out of play)

- How many moves are made by this algorithm?
- M(n) be the number of moves made by the above algorithm
- $M(n) = 2M(n-1) + 1$ with $M(1) = 1$
- 1, 3, 7, 15, 31, ...

Recursive Definitions

- E.g., $f(0) = 1$

  $f(n) = n \cdot f(n-1) \qquad \forall n \in \mathbb{Z}$ s.t. $n>0$

  Initial Condition

  Recurrence relation

- $f(n) = n \cdot (n-1) \cdot \ldots \cdot 1 \cdot 1 = n!$
- This is the formal definition of n!
- Translates to a program to compute factorial:

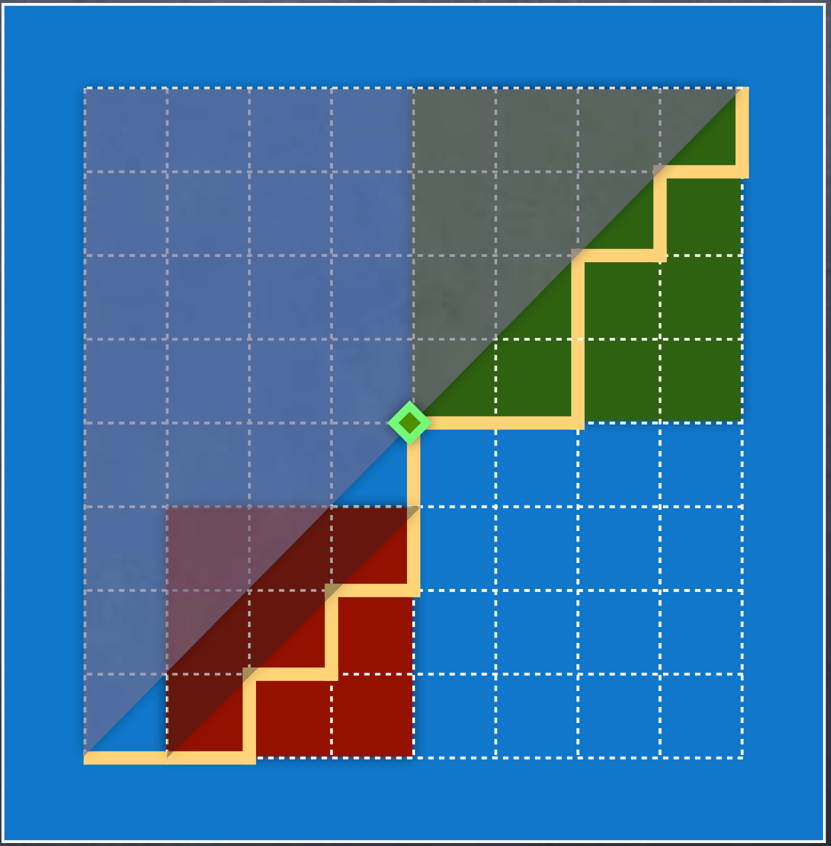
```
factorial(n ∈ ℕ)  {
   if (n==0) return 1;
   else return n*factorial(n-1);
}
```

```
factorial(n ∈ ℕ)  {
   F[0] = 1;
   for i in 1..n
      F[i] = i*F[i-1];
   return F[n];
}
```

Catalan Numbers

- How many paths are there in the grid from (0,0) to (n,n) without ever crossing over to the y>x region?

- Any path can be constructed as follows
  - Pick minimum k>0 s.t. (k,k) reached
  - $(0,0) \rightarrow (1,0) \Rrightarrow (k,k-1) \rightarrow (k,k) \Rrightarrow (n,n)$ where $\Rrightarrow$ denotes a Catalan path



- $\text{Cat}(n) = \Sigma_{k=1 \text{ to } n} \text{Cat}(k-1) \cdot \text{Cat}(n-k)$
- $\text{Cat}(0) = 1$
- 1, 1, 2, 5, 14, 42, 132, ...

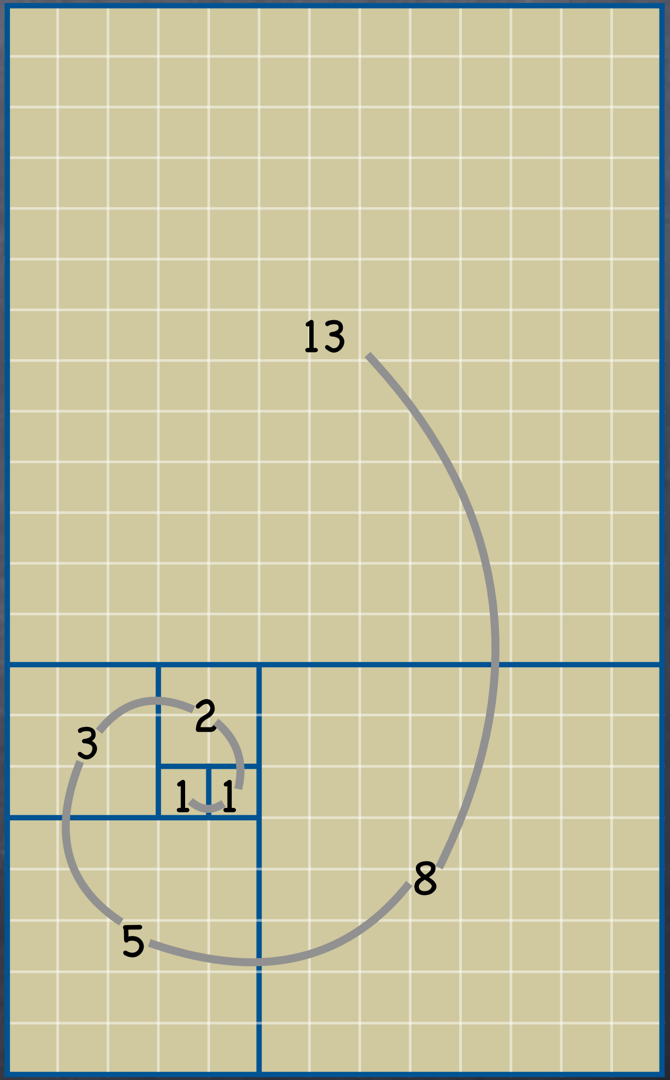
e.g., $42 = 1 \cdot 14 + 1 \cdot 5 + 2 \cdot 2 + 5 \cdot 1 + 14 \cdot 1$

Closed form expression? Later

Fibonacci Sequence

- $F(0) = 0$
  $F(1) = 1$
  $F(n) = F(n-1) + F(n-2) \quad \forall n \geq 2$
- $F(n)$ is the n^{th} Fibonacci number (starting with 0^{th})

Closed form expression? Coming up



Counting Strings

- How many ternary strings of length n which don't have "00" as a substring?
- Set up a recurrence
  - $A(n)$ = # such strings starting with 0
  - $B(n)$ = # such strings not starting with 0
  - $A(n) = B(n-1)$. $B(n) = 2(A(n-1) + B(n-1))$. [Why?]
- Initial condition: $A(0) = 0$; $B(0) = 1$ (empty string)
- Required count: $A(n) + B(n)$
- Can rewrite in terms of just B
  - $B(0) = 1$. $B(1) = 2$. $B(n) = 2B(n-1) + 2B(n-2) \quad \forall n \geq 2$
  - Required count: $B(n-1) + B(n)$.

Recursion & Induction

- Claim: $F(3n)$ is even, where $F(n)$ is the n^{th} Fibonacci number, $\forall n \geq 0$

0 1 1 **2** 3 5 **8** 13 21 **34**...
Stronger claim (but easier to prove by induction): $F(n)$ is even iff n is a multiple of 3

- Proof by induction:
- Base case:
  $n=0$: $F(3n) = F(0) = 0$ ✔ $n=1$: $F(3n) = F(3) = 2$ ✔
- Induction step: for all $k \geq 2$
  Induction hypothesis: suppose for $0 \leq n \leq k-1$, $F(3n)$ is even
  To prove: $F(3k)$ is even
  - $F(3k) = F(3k-1) + F(3k-2) = ?$
    - Unroll further: $F(3k-1) = F(3k-2) + F(3k-3)$
      $F(3k) = 2 \cdot F(3k-2) + F(3(k-1))$ = even, by induction hypothesis

Closed Form

- Sometimes possible to get a "closed form" expression for a quantity defined recursively (in terms of simpler operations)
  - e.g., $f(0)=0$ & $f(n) = f(n-1) + n, \forall n>0$
    - $f(n) = n(n+1)/2$
- Sometimes, we just give it a name
  - e.g., n!, Fibonacci(n), Cat(n)
  - In fact, formal definitions of integers, addition, multiplication etc. are recursive
    - e.g., $0 \cdot a = 0$ & $n \cdot a = (n-1) \cdot a + a, \forall n>0$
    - e.g., $2^0 = 1$ & $2^n = 2 \cdot 2^{n-1}$
- Sometimes both
  - e.g., Fibonacci(n), Cat(n) have closed forms

Closed Form via Induction

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2)$ $\forall n \geq 2$.

Exercise: Fibonacci numbers

- Suppose $X^2 - aX - b = 0$ has two distinct (possibly complex) solutions, x and y

Characteristic equation: replace f(n) by X^n in the recurrence

- Claim: $\exists p,q\ \forall n\ f(n) = p \cdot x^n + q \cdot y^n$
- Let $p=(d-cy)/(x-y)$, $q=(d-cx)/(y-x)$ so that base cases n=0,1 work
- Inductive step: for all $k \geq 2$

  Induction hypothesis: $\forall n$ s.t. $1 \leq n \leq k-1$, $f(n) = px^n + qy^n$

  To prove: $f(k) = px^k + qy^k$

  - $f(k) = a \cdot f(k-1) + b \cdot f(k-2)$

    $= a \cdot (px^{k-1}+qy^{k-1}) + b \cdot (px^{k-2}+qy^{k-2}) - px^k - qy^k + px^k + qy^k$

    $= - px^{k-2}(x^2-ax-b) - qy^{k-2}(y^2-ay-b) + px^k + qy^k = px^k + qy^k$ ✓

Closed Form via Induction

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2) \quad \forall n \geq 2$.
- Suppose $X^2 - aX - b = 0$ has only one solution $x \neq 0$
  i.e., $X^2 - aX - b = (X-x)^2$, or equivalently, $a=2x$, $b=-x^2$
- Claim: $\exists p,q \ \forall n \ f(n) = (p + q \cdot n)x^n$
- Let $p = c$, $q = d/x-c$ so that base cases $n=0,1$ work
- Inductive step: for all $k \geq 2$
  Induction hypothesis: $\forall n$ s.t. $1 \leq n \leq k-1$, $f(n) = (p + qn)y^n$
  To prove: $f(k) = (p+qk)x^k$
  - $f(k) = a \cdot f(k-1) + b \cdot f(k-2)$
    $= a\,(p+qk-q)x^{k-1} + b \cdot (p+qk-2q)x^{k-2} - (p+qk)x^k + (p+qk)x^k$
    $= -(p+qk)x^{k-2}(x^2-ax-b) - qx^{k-2}(ax+2b) + (p+qk)x^k = (p+qk)x^k$ ✓

Solving a Recurrence

- Often, once a correct guess is made, easy to prove by induction
- How does one guess?
- Will see a couple of approaches
  - By unrolling the recurrence into a chain or a "rooted tree"
  - Using the "method of generating functions"

Recursive Definitions

Unrolling Recurrences

Unrolling a recursion

- Often helpful to try "unrolling" a recursion to see what is happening
- e.g., expand into a chain:
  - $T(0) = 0$ & $T(n) = T(n-1) + n^2$ $\forall n \geq 1$
    - $T(n-1) = T(n-2) + (n-1)^2$, $T(n-2) = T(n-3) + (n-2)^2$, ...
    - $T(n) = n^2 + (n-1)^2 + (n-2)^2 + T(n-3)$ $\forall n \geq 3$
    - $T(n) = \Sigma_{k=1 \text{ to } n}\ k^2 + T(0)$ $\forall n \geq 0$

Another example

- $T(1) = 0$
  $T(N) = T(\lfloor N/2 \rfloor) + 1 \quad \forall N \geq 2$
- Let us consider N of the form 2^n (so we can forget the floor)
- $T(N) = 1 + T(N/2)$
  $= 1 + 1 + T(N/4)$
  $= \ldots$
  $= 1 + 1 + \ldots + T(1)$

  How many 1's are there?

- $T(2^n) = n$
- $T(N) = \log_2 N$ (or simply $\log N$) for N a power of 2

  A slowly growing function

- General N? T monotonically increasing (by strong induction). So,
  $T(2^{\lfloor \log N \rfloor}) \leq T(N) \leq T(2^{\lceil \log N \rceil})$: i.e., $\lfloor \log N \rfloor \leq T(N) \leq \lceil \log N \rceil$
  - In fact, $T(N) = \lfloor \log N \rfloor$ (Exercise)

Tower of Hanoi

- Recursive algorithm (optimal for 3 pegs)
  - Transfer(n,A,C):

    If n=1, move the single disk from peg A to peg C

    Else

      Transfer(n-1,A,B) (leaving the largest disk out of play)

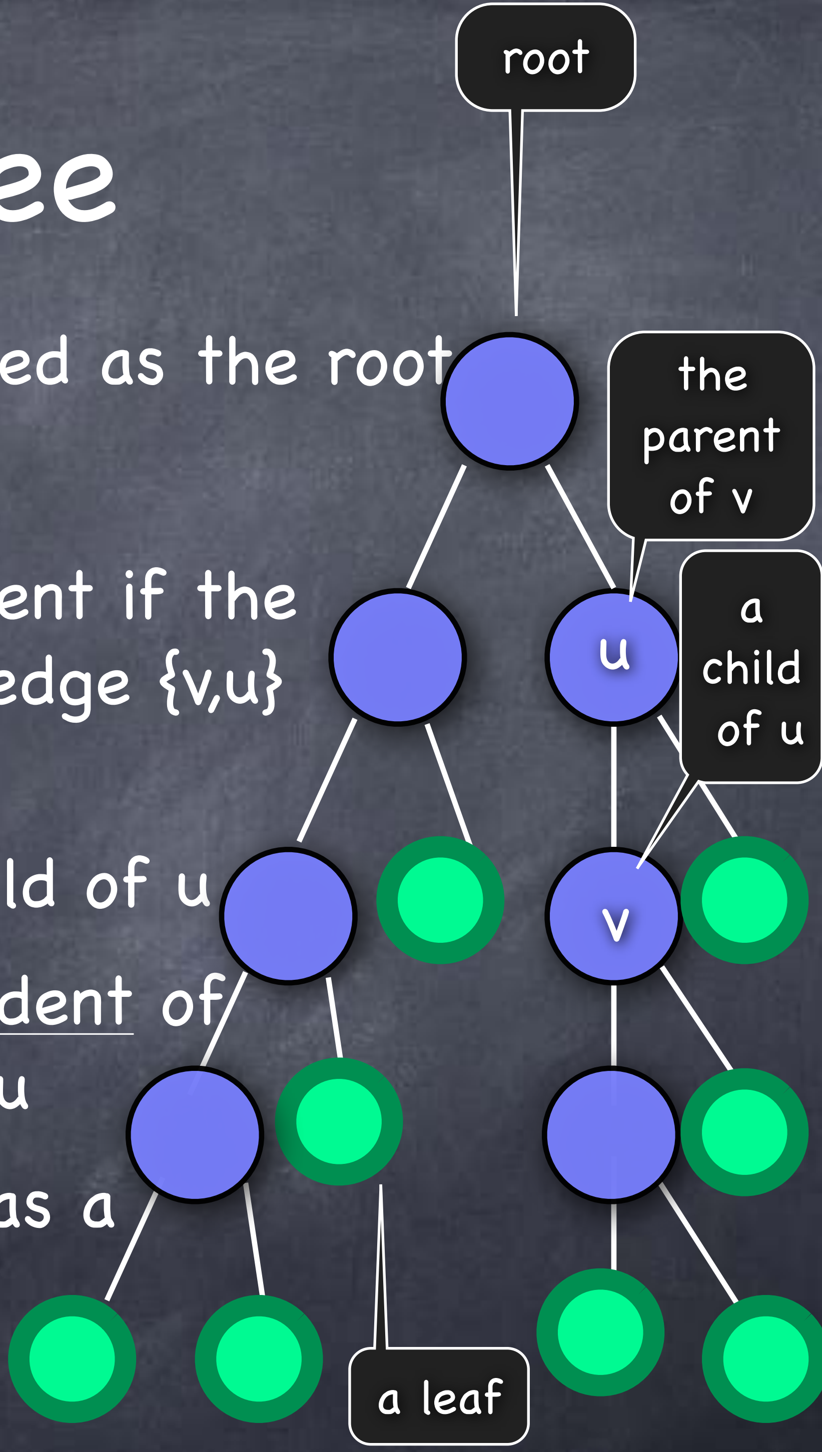
      Move largest disk to peg C

      Transfer(n-1,B,C) (leaving the largest disk out of play)

- $M(n)$ be the number of moves made by the above algorithm
- $M(n) = 2M(n-1) + 1$ with $M(1) = 1$
- Unroll the recursion into a "rooted tree"

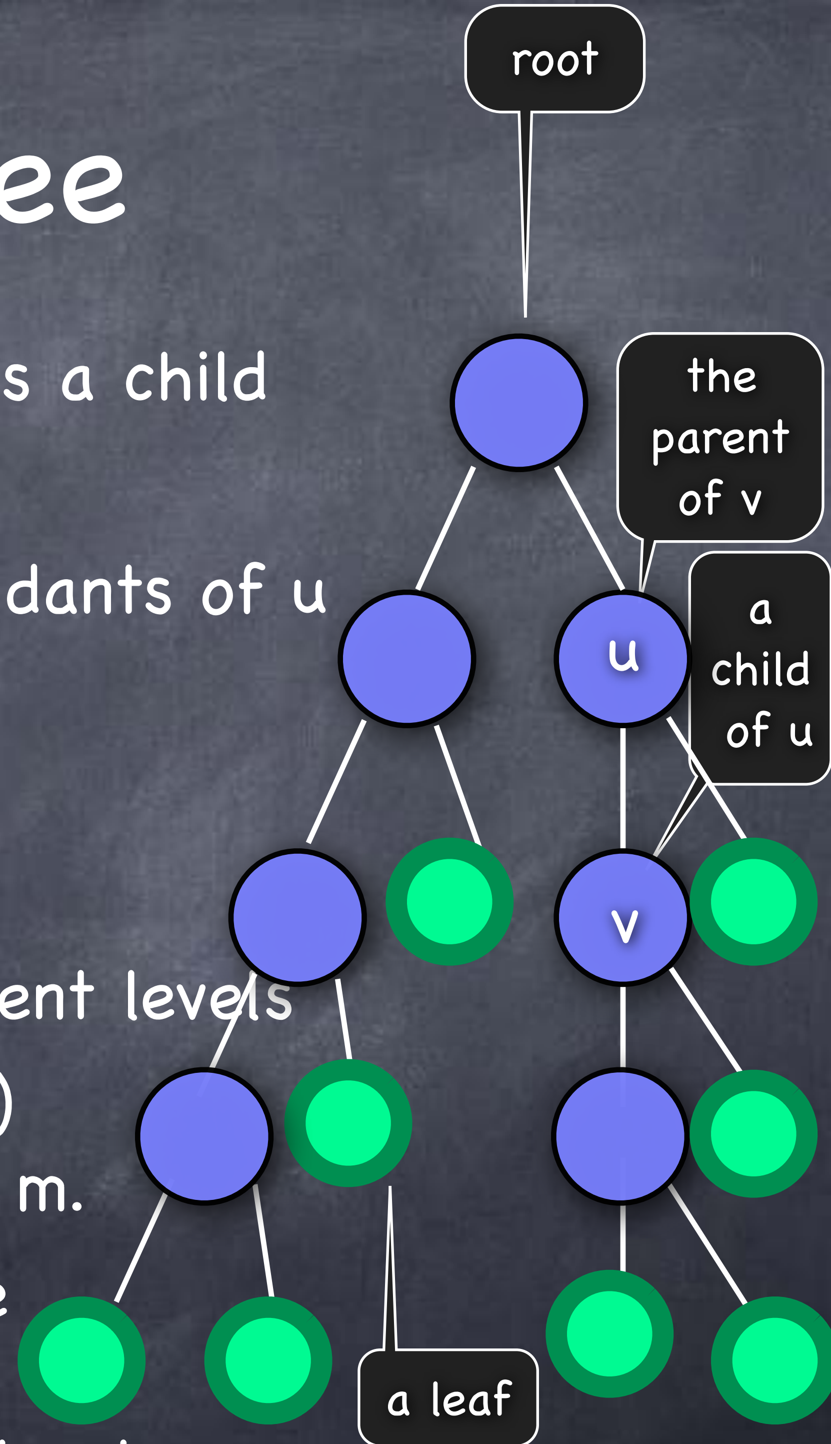
Rooted Tree

- A tree, with a special node, designated as the root
- Typically drawn "upside down"
- Parent and child relation: u is v's parent if the unique path from v to root contains edge {v,u} (parent unique; root has no parent)
    - If u is v's parent v, then v is a child of u
- u is an ancestor of v, and v a descendent of u if the v-root path passes through u
- Leaf is redefined for a rooted tree, as a node with no child
    - Root is a leaf iff it has degree 0 (if deg(root)=1, conventionally not called a leaf)



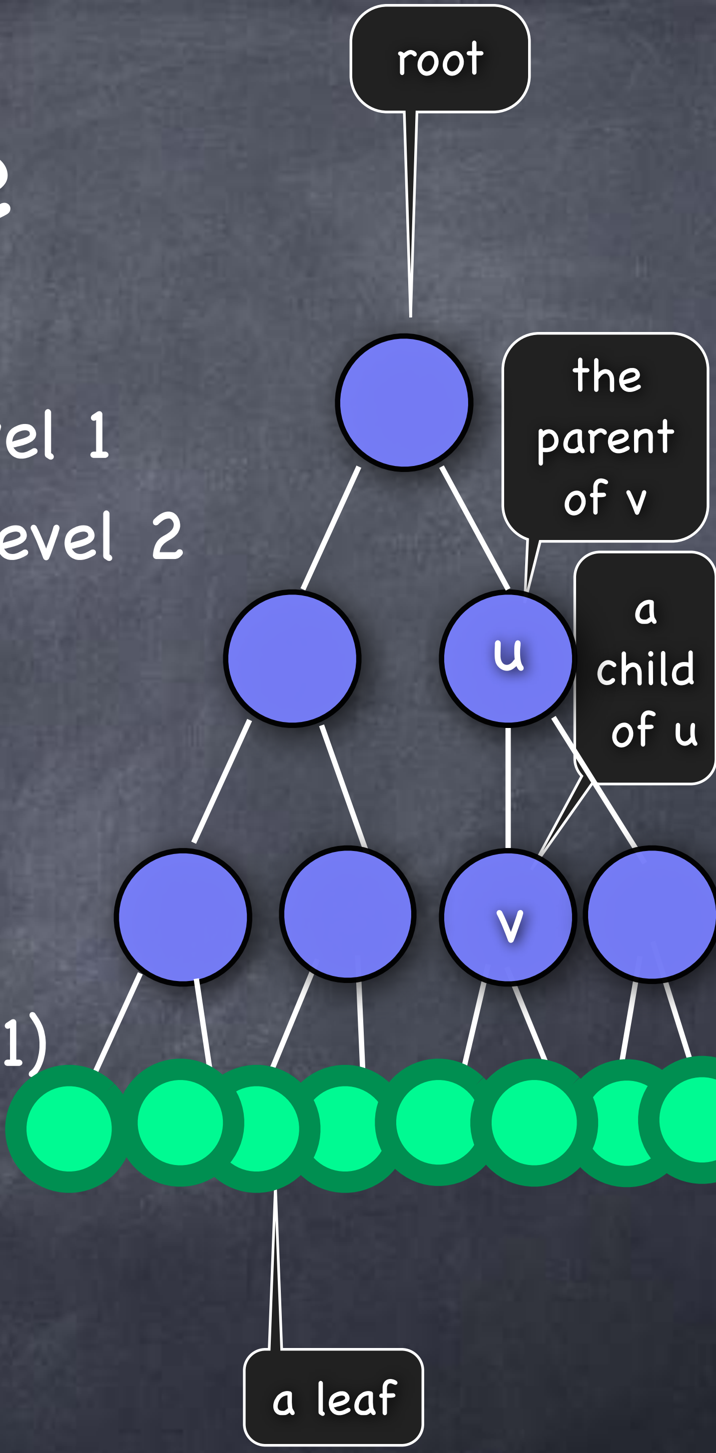
Rooted Tree

- Leaf: no children. Internal node: has a child
- Ancestor, descendant: partial orders
- Subtree rooted at u: with all descendants of u
- Depth of a node: distance from root.
  Height of a tree: maximum depth
- Level i: Set of nodes at depth i.
- Note: tree edges are between adjacent levels
- Arity of a tree: Max (over all nodes) number of children. m-ary if arity $\leq$ m.
- Full m-ary tree: Every internal node has exactly m children.
  Complete & Full: All leaves at same level



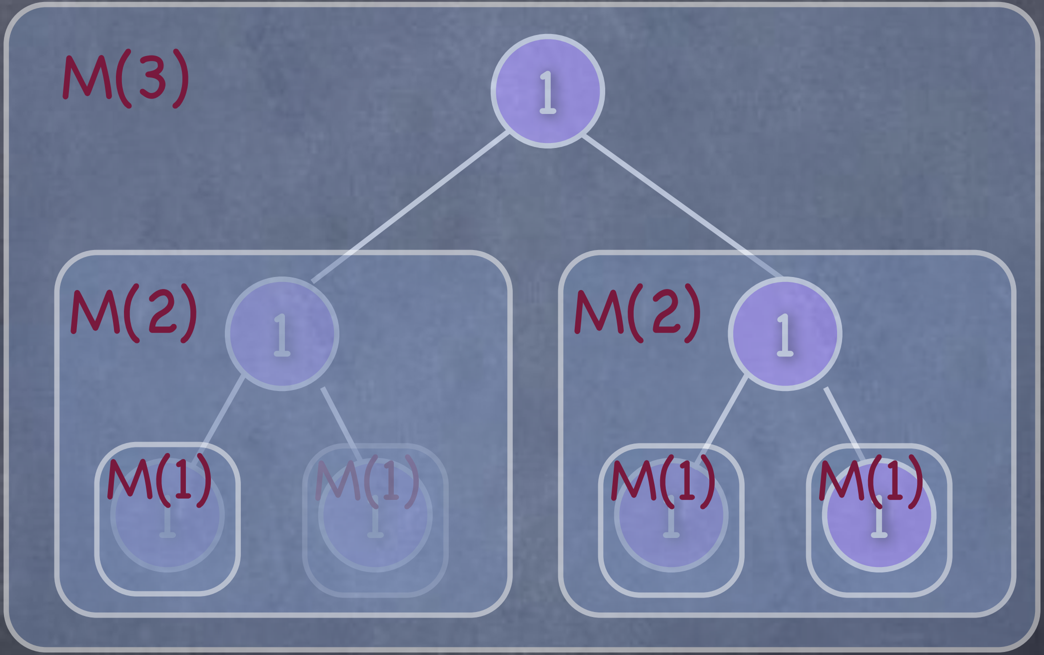
Rooted Tree

- Complete & Full m-ary tree
  - One root node with m children at level 1
  - Each level 1 node has m children at level 2
    - m^2 nodes at level 2
  - At level i, m^i nodes
  - m^h leaves, where h is the height
- Total number of nodes:
  - $m^0 + m^1 + m^2 + \ldots + m^h = (m^{h+1}-1)/(m-1)$
    - Prove by induction:
      $(m^h-1)/(m-1) + m^h = (m^{h+1}-1)/(m-1)$
- Binary tree (m=2)
  - 2^h leaves, 2^h-1 internal nodes



Tower of Hanoi

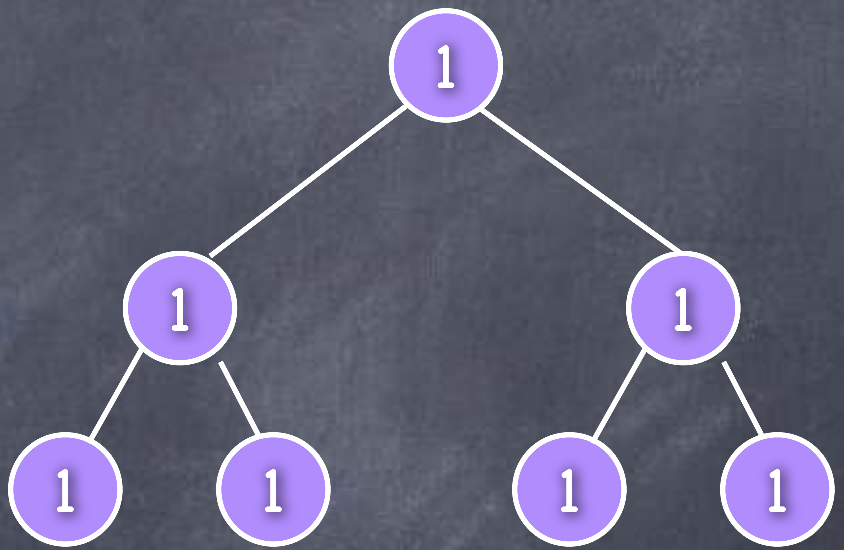
- $M(1) = 1$
  $M(n) = 2M(n-1) + 1$



Doing it bottom-up. Could also think top-down

Tower of Hanoi

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



- Exponential growth
- $M(2) = 3$, $M(3) = 7$, ...
- $M(n)$ = #nodes in a complete and full binary tree of height $n-1$
- $M(n) = 2^n - 1$

Recursive Definitions

Generating Functions

Generating Functions

- A generating function is an alternate representation of an infinite sequence, which allows making useful deductions about the sequence (including, possibly, a closed form)

- Sequence $f(0), f(1), \ldots$ is represented as the formal expression $G_f(X) \triangleq f(0) + f(1)\cdot X + f(2)\cdot X^2 + \ldots$ (ad infinitum)

    - i.e., for $f : \mathbb{N} \rightarrow \mathbb{R}$, we define $G_f(X) \triangleq \Sigma_{k\geq 0}\, f(k)\cdot X^k$

- e.g., If $f(k) = a^k$ for some $a \in \mathbb{R}$, $G_f(X) = \Sigma_{k\geq 0}\, a^k \cdot X^k$

"Ordinary Generating Functions"

Generating Functions

- Generating functions sometimes have a succinct representation
- e.g., For $f(k) = a^k$ for some $a \in \mathbb{R}$, $G_f(X) = \Sigma_{k \geq 0}\, a^k \cdot X^k$
  - If we substituted for X a real number x sufficiently close to 0, we have $|ax| < 1$ and this would converge to $1/(1-ax)$
  - So we write $G_f(X) = 1/(1-aX)$ (for sufficiently small $|X|$). This will later let us manipulate $G_f(X)$ algebraically

Extended Binomial Theorem

- A useful tool for manipulating/analysing generating functions
- For $a \in \mathbb{R}$, $\binom{a}{k} \triangleq \dfrac{a(a-1)\ldots(a-k+1)}{k!}$ $(k \in \mathbb{Z}^+)$, and $\binom{a}{0} \triangleq 1$
- Extended binomial theorem:

  For $|x|<1$, $a \in \mathbb{R}$, $(1+x)^a = \Sigma_{k\geq 0} \binom{a}{k} \cdot x^k$

- Useful in finding a closed form for f given G_f of certain forms
- e.g., $G_f(X) = 1/(1-X)$. Then, $\Sigma_{k\geq 0}\, f(k) \cdot X^k = (1-X)^{-1}$
  - $\binom{-1}{k} = (-1)(-2)\ldots(-k)/k! = (-1)^k \Rightarrow (1-X)^{-1} = \Sigma_{k\geq 0}\, X^k \Rightarrow f(k)=1$
- Similarly, $\binom{-2}{k} = (-2)(-3)\ldots(-k-1)/k! = (-1)^k(k+1)$

  $\Rightarrow 1/(1-X)^2 = \Sigma_{k\geq 0}\, (k+1) \cdot X^k$

Extended Binomial Theorem

- $G_f(X) = 1/(1-aX)^b$ for $f(k) = (-a)^k \cdot \binom{-b}{k} = \binom{b+k-1}{k} \cdot a^k$
  - e.g., b=1: $f(k) = a^k$.   b=2: $f(k) = (k+1).a^k$
- $G_{f+g}(X) = G_f(X) + G_g(X)$
- $G_g(X) = X \cdot G_f(X)$, where $g(0)=0$ and $g(k+1) = f(k)$ for $k \geq 0$
- If a generating function G_f is known and has a nice form, then often using the extended binomial theorem, one can compute a closed-form expression for f
- But how do we get G_f ?

Generating Functions From Recurrence Relations

- e.g., $f(0)=0$, $f(1) = 1$. $f(n) = f(n-1) + f(n-2)$, $\forall n \geq 2$. [Fibonacci]
- $f(n) \cdot X^n = X \cdot f(n-1) \cdot X^{n-1} + X^2 \cdot f(n-2) \cdot X^{n-2}$ (for $n \geq 2$)
  $\Rightarrow \sum_{n\geq 2} f(n) \cdot X^n = X \cdot \sum_{n\geq 2} f(n-1) \cdot X^{n-1} + X^2 \cdot \sum_{n\geq 2} f(n-2) \cdot X^{n-2}$
  $\Rightarrow G_f(X) - f(0) - f(1) \cdot X = X \cdot (G_f(X) - f(0)) + X^2 \cdot G_f(X)$
  $\Rightarrow G_f(X)\,(1-X-X^2) = f(0) + (f(1)-f(0)) \cdot X$
  - $G_f(X) = X/(1-X-X^2)$
- More generally:
  $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2)$, $\forall n \geq 2$
  - $G_f(X) = (c + (d-ac)X)/(1-aX-bX^2)$

Generating Functions For Series Summation

- Suppose $g(k) = \Sigma_{j=0 \text{ to } k}\ f(j)$
- What is $G_g(X)$, in terms of $G_f(X)$?
  - Recursive definition: $g(0) = f(0)$. $g(n) = g(n-1) + f(n)$, $\forall n \geq 1$.
  - So, $\forall k \geq 1$, $g(k) \cdot X^k = g(k-1) \cdot X^{k-1} \cdot X + f(k) \cdot X^k$
  - $G_g(X) = g(0) + X \cdot G_g(X) + (G_f(X) - f(0))$
  - **$G_g(X) = G_f(X)/(1-X)$**

Recursive Definitions

Generating Functions

More Examples

Recall

Generating Functions

- For $f : \mathbb{N} \to \mathbb{R}$, we defined $G_f(X) \triangleq \Sigma_{k\geq 0} f(k)\cdot X^k$
- The extended binomial theorem
  - e.g., $G_f(X) = 1/(1-aX)^b$ for $f(k) = (-a)^k \cdot \binom{-b}{k}$

$$= \binom{b+k-1}{k} \cdot a^k, \text{ for } b \in \mathbb{Z}^+$$

- Combinations: e.g., $G_h(X) = G_f(X) + G_g(X)$, where $h(k)=f(k)+g(k)$

  $G_g(X) = \alpha X\, G_f(X)$, where $g(0) = 0$, $g(k) = \alpha f(k-1)\ \forall k > 0$

  $G_h(X) = (1+\alpha X)\, G_f(X)$, where $h(0)=f(0)$, $h(k) = f(k) + \alpha f(k-1)\ \forall k>0$
- From recurrence relations
  - e.g., If $f(0) = c$. $f(1) = d$. $f(n) = a\cdot f(n-1) + b\cdot f(n-2)$, $\forall n\geq 2$
    - $G_f(X) = (c + (d-ac)X)/(1-aX-bX^2)$
  - e.g., If $g(k) = \Sigma_{j=0 \text{ to } k}\ f(j)$
    - $G_g(X) = G_f(X)/(1-X)$

Generating Functions For Series Summation

- e.g., $g(k) = \Sigma_{j=0 \text{ to } k} (j+1)^2$
- $G_g(X) = G_f(X)/(1-X)$ where $f(j) = (j+1)^2$
- Consider $G(X) = 1 + X + X^2 + \dots = 1/(1-X)$
  - $G'(X) = 1 + 2 \cdot X + 3 \cdot X^2 + \dots = 1/(1-X)^2$
  - Let $H(X) = X\,G(X) = X + 2 \cdot X^2 + 3 \cdot X^3 + \dots = X/(1-X)^2$
  - So $H'(X) = 1 + 2^2 \cdot X + 3^2 \cdot X^2 + \dots = 1/(1-X)^2 + 2X/(1-X)^3$
    $= (1+X)/(1-X)^3$

    is the generating function of $f(j) = (j+1)^2$.
- $G_g(X) = (1+X)/(1-X)^4$.
- Exercise: use ext. binomial theorem to compute coeff. of X^k

Calculus!

Alternately, from extended binomial theorem

Generating Functions For Counting Combinations

- e.g., Let $f(n)$ = number of ways to throw n unlabelled balls into d labelled bins (for some fixed number d)
  - Solution 1: Use stars and bars
  - Solution 2: Reason about $G_f(X)$
    - Coefficient of X^n in $G_f(X)$ must count the number of (non-negative integer) solutions of $n_1 + \ldots + n_d = n$
    - Can write $G_f(X) = (1+X+X^2+\ldots)^d$
    - So, $G_f(X) = [1/(1-X)]^d = (1-X)^{-d}$
    - Coefficient of X^n $= \binom{-d}{n}(-1)^n$

$$= d\,(d+1) \ldots (d+n-1) \,/\, n! = C(d+n-1,n)$$

Recall

A Closed Form

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2)$ $\forall n \geq 2$.
- Suppose $X^2 - aX - b = 0$ has two distinct (possibly complex) solutions, x and y
- Claim: **$\exists p,q\ \forall n\ f(n) = p \cdot x^n + q \cdot y^n$**
- Let $p=(d-cy)/(x-y)$, $q=(d-cx)/(y-x)$ so that base cases n=0,1 work
- Inductive step: for all $k \geq 2$
  Induction hypothesis: $\forall n$ s.t. $1 \leq n \leq k-1$, $f(n) = px^n + qy^n$
  To prove: $f(k) = px^k + qy^k$
  - $f(k) = a \cdot f(k-1) + b \cdot f(k-2)$
    $= a \cdot (px^{k-1}+qy^{k-1}) + b \cdot (px^{k-2}+qy^{k-2}) - px^k - qy^k + px^k + qy^k$
    $= - px^{k-2}(x^2-ax-b) - qy^{k-2}(y^2-ay-b) + px^k + qy^k = px^k + qy^k$ ✓

Recall

A Closed Form

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2) \quad \forall n \geq 2$.
- Suppose $X^2 - aX - b = 0$ has only one solution $x \neq 0$
  i.e., $X^2 - aX - b = (X-x)^2$, or equivalently, $a=2x$, $b=-x^2$
- Claim: **$\exists p,q \; \forall n \; f(n) = (p + q \cdot n)x^n$**
- Let $p = c$, $q = d/x-c$ so that base cases $n=0,1$ work
- Inductive step: for all $k \geq 2$
  Induction hypothesis: $\forall n$ s.t. $1 \leq n \leq k-1$, $f(n) = (p + qn)y^n$
  To prove: $f(k) = (p+qk)x^k$
  - $f(k) = a \cdot f(k-1) + b \cdot f(k-2)$
    $= a\,(p+qk-q)x^{k-1} + b \cdot (p+qk-2q)x^{k-2} - (p+qk)x^k + (p+qk)x^k$
    $= -(p+qk)x^{k-2}(x^2-ax-b) - qx^{k-2}(ax+2b) + (p+qk)x^k = (p+qk)x^k$ ✓

A Closed Form

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2) \quad \forall n \geq 2$.
- Recall: $G_f(X) = (c + (d-ac)X)/(1-aX-bX^2)$
- Let $G_f(X) = (\alpha + \beta X)/(1-aX-bX^2)$. i.e., $\alpha = c$, $\beta = d-ac$.
- Writing $Z = X^{-1}$, we have $G_f(X) = (\alpha Z^2 + \beta Z)/(Z^2-aZ-b)$
- Let $(Z^2-aZ-b) = (Z-x)(Z-y)$
  - $a = x+y$, $-b = xy$
  - $(1-aX-bX^2) = (1-xX)(1-yX)$
- Two cases: $x \neq y$ and $x = y$

A Closed Form

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2) \quad \forall n \geq 2$.
- $G_f(X) = (\alpha + \beta X)/[(1-xX)(1-yX),]$ where $\alpha = c$, $\beta = d-ac$, $a = x+y$, $-b = xy$.
- Case 1: $x \neq y$.
  - $1/[(1-xX)(1-yX)] = [\ 1/(1-xX) - 1/(1-yX)\ ]/[X(x-y)]$
  - Recall, $1/(1-xX) = \Sigma_{k\geq 0} (xX)^k$
  - So, $G_f(X) = (\alpha/X + \beta)/(x-y) \cdot \Sigma_{k\geq 0} (xX)^k - (yX)^k$
    $= \Sigma_{k\geq 1} \alpha(x\cdot(xX)^{k-1} - y\cdot(yX)^{k-1})/(x-y) + \Sigma_{k\geq 0} \beta((xX)^k-(yX)^k)/(x-y)$
    $= \Sigma_{k\geq 0} (px^k+qy^k)\cdot X^k$, where $p = (\alpha x+\beta)/(x-y)$, $q = (\alpha y+\beta)/(y-x)$
  - $f(n) =$ coefficient of $X^n = px^n + qy^n$
- $\alpha = c$, $\beta = d-ac = d-(x+y)c \Rightarrow\ p = (d-yc)/(x-y)$, $q = (d-xc)/(y-x)$,

A Closed Form

- $f(0) = c$. $f(1) = d$. $f(n) = a \cdot f(n-1) + b \cdot f(n-2)$ $\forall n \geq 2$.
- $G_f(X) = (\alpha + \beta X)/[(1-xX)(1-yX),]$ where $\alpha = c$, $\beta = d - ac$, $a = x+y$, $-b = xy$.
- Case 2: $x = y \neq 0$.
  - $G_f(X) = (\alpha + \beta X)/(1-xX)^2$
  - Recall, $1/(1-xX)^2 = \Sigma_{k\geq 0} (k+1).x^k \cdot X^k$
  - $(\alpha + \beta X)/(1-xX)^2 = \Sigma_{k\geq 0} (\alpha + \beta X)\cdot(k+1)\cdot x^k \cdot X^k$

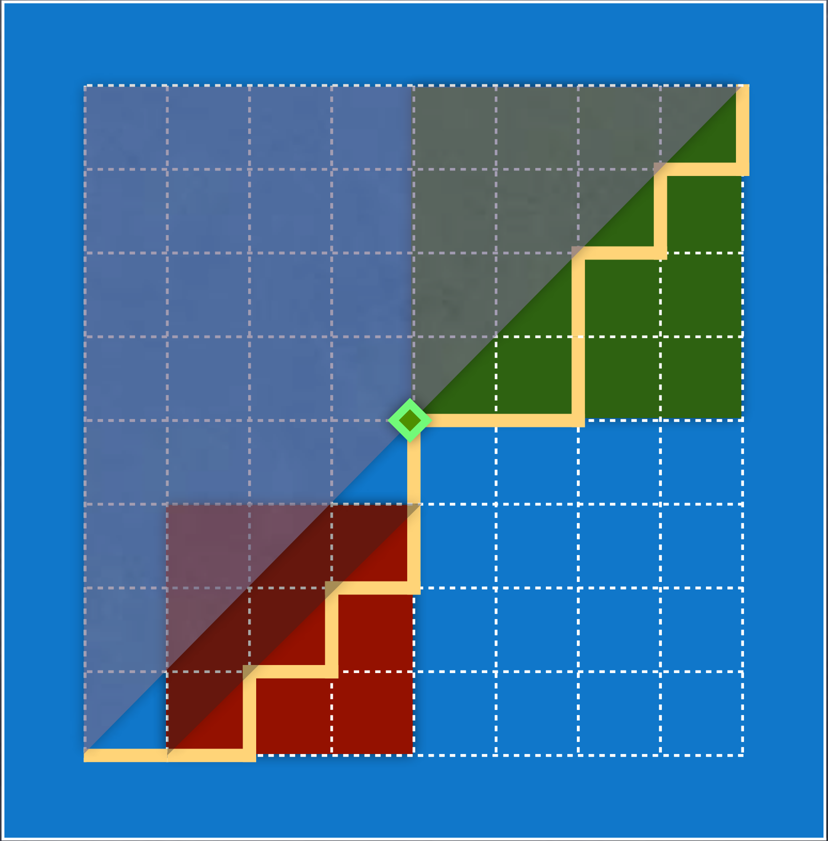
    $= \Sigma_{k\geq 0} (\alpha\cdot(k+1)\cdot x^k + \beta\cdot k \cdot x^{k-1})\cdot X^k$

    $= \Sigma_{k\geq 0} (p + qk)x^k \cdot X^k$, where $p = \alpha$, $q = (\alpha + \beta/x)$

Recall

Catalan Numbers

- How many paths are there in the grid from (0,0) to (n,n) without ever crossing over to the y>x region?
- Any path can be constructed as follows
  - Pick minimum k>0 s.t. (k,k) reached
  - $(0,0) \rightarrow (1,0) \Rrightarrow (k,k-1) \rightarrow (k,k) \Rrightarrow (n,n)$ where $\Rrightarrow$ denotes a Catalan path
- $\mathrm{Cat}(n) = \Sigma_{k=1 \text{ to } n} \mathrm{Cat}(k-1) \cdot \mathrm{Cat}(n-k)$
- $\mathrm{Cat}(0) = 1$
- 1, 1, 2, 5, 14, 42, 132, ...



e.g., $42 = 1 \cdot 14 + 1 \cdot 5 + 2 \cdot 2 + 5 \cdot 1 + 14 \cdot 1$

Closed form expression?

Catalan Numbers

- $Cat(n) = \Sigma_{k=1 \text{ to } n} Cat(k-1) \cdot Cat(n-k) \quad \forall n \geq 1$
- $Cat(n)\ X^n = \Sigma_{k=1 \text{ to } n} Cat(k-1) \cdot Cat(n-k) \cdot X^n$
  $= \text{term of } X^n \text{ in } X \cdot (\Sigma_{k \geq 1} Cat(k-1)\ X^{k-1}) \cdot (\Sigma_{k \leq n} Cat(n-k)\ X^{n-k}), \quad \forall n \geq 1$
  - For $n=0$, we have $Cat(0)\ X^0 = 1$
- $G_{Cat}(X) = 1 + X\ G_{Cat}(X)\ G_{Cat}(X)$
- Solving for G in $X \cdot G^2 - G + 1 = 0$, we have $G=[1 \pm \sqrt{(1-4X)}]/(2X)$
  - We need $\lim_{X \to 0} G_{cat}(X) = Cat(0) = 1$
    - $\lim_{X \to 0} [1 \pm \sqrt{(1-4X)}]/(2X) = \lim_{X \to 0} \pm (-4/[2\sqrt{(1-4X)}])/2 = \pm(-1)$

L'Hôpital's Rule

  - So we take $G_{cat}(X) = [1-\sqrt{(1-4X)}]/(2X)$
  - Then, what is the coefficient of X^n in $G_{cat}(X)$?

Catalan Numbers

- $G_{cat}(X) = [1-\sqrt{(1-4X)}]/(2X)$
- Then, what is the coefficient of X^k in $G_{cat}(X)$?
- Use extended binomial theorem:

$$(1-4X)^{½} = \Sigma_{k\geq 0} \binom{1/2}{k} (-4X)^k = 1 + \Sigma_{k\geq 1} -2\binom{2(k-1)}{k-1}/k \cdot X^k$$

- for $k>0$, $\binom{1/2}{k} = (1/2)(-1/2)\,(-3/2)\,(-5/2)\,\ldots\,(-(2k-3)/2)\,/k!$

$$= (-1)^{k-1}(1\cdot 1\cdot 3\cdot \ldots\cdot(2k-3))/[k!\ 2^k] = (-1)^{k-1}\binom{2k-2}{k-1}/[k\ 2^{2k-1}]$$

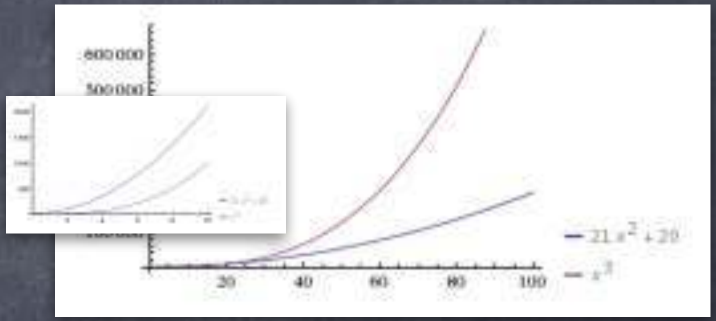
$2 \cdot 4 \cdot \ldots \cdot (2k-2)$

$(k-1)! \cdot 2^{k-1}$

- $G_{cat}(X) = \Sigma_{k\geq 1} \binom{2(k-1)}{k-1}/k \cdot X^{k-1}$
- $Cat(k)$ = Coefficient of X^k in $G_{cat}(X)$ $= \binom{2k}{k}/(k+1)$

Asymptotics

The Big O



How it scales

- In analysing running time (or memory/power consumption) of an algorithm, we are interested in how it <u>scales</u> as the problem instance grows in “size”
  - Running time on small instances of a problem are often not a serious concern (anyway small)
- Also, exact time/number of steps is less interesting
  - Can differ in different platforms. Not a property of the algorithm alone.
  - Thus “unit of time” (constant factors) typically ignored when analysing the algorithm.

How it scales

- e.g., suppose number of "steps" taken by an algorithm to sort a list of n elements varies between 3n and $3n^2+9$ (depending on what the list looks like)
    - If n is doubled, time taken in the worst case could become (roughly) 4 times. If n is tripled, it could become (roughly, in the worst case) 9 times
    - An upper bound that grows "like" n^2
- Typically, interested in easy to interpret guarantees
    - Resource required expressed as a function of input size
    - Upper bounds robust to constant factor speed ups

Upper-bounds: Big O

- T(n) has an upper-bound that grows "like" f(n)
  - $T(n) = O(f(n))$

    $\exists c, k > 0, \ \ \forall n \geq k, \ \ 0 \leq T(n) \leq c \cdot f(n)$

    Unfortunate notation! An alternative used sometimes: $T(n) \in O(f(n))$

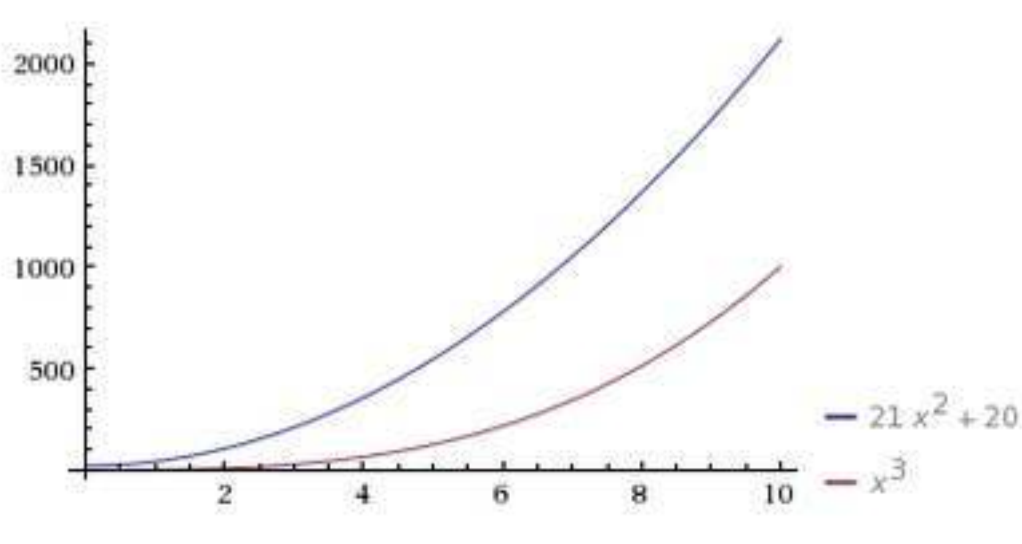
  - Note: we are defining it only for T & f which are eventually non-negative
  - Note: order of quantifiers! c can't depend on n (that is why c is called a constant factor)
- Important: If $T(n)=O(f(n))$, f(n) could be much larger than T(n) (but only a constant factor smaller than T(n))

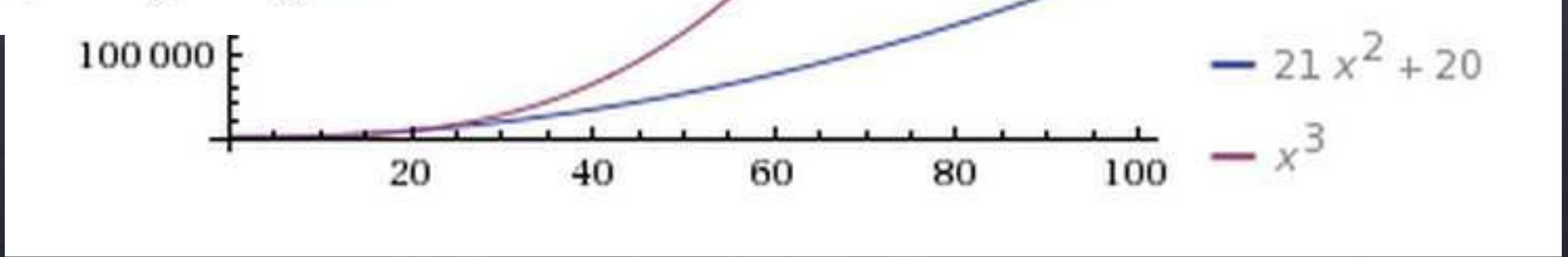
$T(n) = O(f(n))$

$\exists c, k > 0, \quad \forall n \geq k, \quad 0 \leq T(n) \leq c \cdot f(n)$

Upper-bounds: Big O

- e.g. $T(x) = 21x^2 + 20$
- $T(x) = O(x^3)$





$T(n) = O(f(n))$
$\exists c, k > 0, \ \ \forall n \geq k, \ \ 0 \leq T(n) \leq c \cdot f(n)$

Upper-bounds: Big O

- e.g. $T(x) = 21x^2 + 20$
- $T(x) = O(x^3)$
- $T(x) = O(x^2)$ too, since we allow scaling by constants
- But $T(x) \neq O(x)$.
  - $\forall c>0, \ \forall k>0, \ \exists x^* \geq k \ \ T(x^*) > c.x^*$

$T(n) = O(f(n))$
$\exists c, k > 0, \quad \forall n \geq k, \quad 0 \leq T(n) \leq c \cdot f(n)$

Upper-bounds: Big O

- Used in the analysis of running time of algorithms: Worst-case Time(input size) = O(f(input size))
  - e.g. $T(n) = O(n^2)$, $T(n) = O(n \log n)$
- Also used to bound approximation errors
  - e.g., $|\log(n!) - \log(n^n)| = O(n)$
    - A better approximation: $|\log(n!) - \log((n/e)^n)| = O(\log n)$
    - Even better: $|\log(n!) - \log((n/e)^n) - ½ \cdot \log(n)| = O(1)$
- We may also have $T(n) = O(f(n))$, where f is a decreasing function (especially when bounding errors)
  - e.g. $T(n) = O(1/n)$

$T(n) = O(f(n))$
$\exists c, k > 0, \quad \forall n \geq k, \quad 0 \leq T(n) \leq c \cdot f(n)$

Big O: Some Properties

- Suppose $T(n) = O(f(n))$ and $R(n) = O(f(n))$
  - i.e., $\forall n \geq k_T,\ 0 \leq T(n) \leq c_T \cdot f(n)$ and $\forall n \geq k_R,\ 0 \leq R(n) \leq c_R \cdot f(n)$
  - $T(n) + R(n) = O(f(n))$
    - Then, $\forall n \geq \max(k_T, k_R),\ \ 0 \leq T(n)+R(n) \leq (c_R+c_T) \cdot f(n)$
  - If eventually ($\forall n \geq k$), $R(n) \leq T(n)$, then $T(n) - R(n) = O(T(n))$
    - $\forall n \geq \max(k, k_R),\ 0 \leq T(n) - R(n) \leq 1 \cdot T(n)$
- If $T(n) = O(g(n))$ and $g(n) = O(f(n))$, then $T(n) = O(f(n))$
    - $\forall n \geq \max(k_T, k_g),\ 0 \leq T(n) \leq c_T \cdot g(n) \leq c_T c_g \cdot f(n)$
- e.g., $7n^2 + 14n + 2 = O(n^2)$ because $7n^2$, $14n$, 2 are all $O(n^2)$
- More generally, if $T(n)$ is upper-bounded by a degree d polynomial with a positive coefficient for n^d, then $T(n) = O(n^d)$

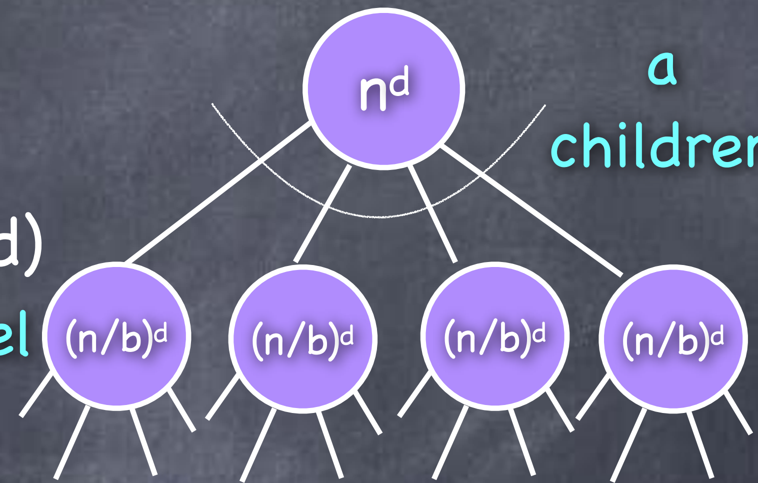
$T(n) = O(f(n))$
$\exists c, k > 0, \quad \forall n \geq k, \quad 0 \leq T(n) \leq c \cdot f(n)$

Some important functions

- $T(n) = O(1)$: $\exists c$ s.t. $T(n) \leq c$ for all ~~sufficiently large~~ n
- $T(n) = O(\log n)$. $T(n)$ grows quite slowly, because log n grows quite slowly (when n doubles, log n grows by 1)
- $T(n) = O(n)$: $T(n)$ is (at most) linear in n
- $T(n) = O(n^2)$: $T(n)$ is (at most) quadratic in n
- $T(n) = O(n^d)$ for some fixed d: $T(n)$ is (at most) polynomial in n
- $T(n) = O(2^{d \cdot n})$ for some fixed d: $T(n)$ is (at most) exponential in n. $T(n)$ could grow very quickly.

A General Solution (a.k.a. "Master Theorem")

- $T(n) = a\,T(n/b) + c \cdot n^d$ (and $T(1)=1$. $a \geq 1, b > 1$ integer, $c > 0$, $d \geq 0$ real.)



- Say $n = b^k$ (so only integers encountered)
- #levels = $\log_b n = k$

total at this level $= a \cdot (n/b)^d$

- $T(n) = O(\, n^d \,(\, 1 + (a/b^d) + \ldots + (a/b^d)^k \,)$ total at i^{th} level $= a^i \cdot (n/b^i)^d$
- If $a = b^d$, contribution at each level $= n^d$. $T(n) = O(n^d \cdot \log n)$
- If $a < b^d$: $1 + (a/b^d) + (a/b^d)^2 + \ldots = O(1)$. $T(n) = O(n^d)$
- If $a > b^d$: $(a/b^d)^k[1 + (b^d/a) + (b^d/a)^2 + \ldots\,] = O((a/b^d)^k) = a^k/n^d$
  $T(n) = O(a^k) = O(2^{k \cdot \log a}) = O(2^{\log n \,\cdot\, \log a / \log b}) = O(n^{\log_b a})$

Tight Bounds: Theta Notation

- If we can give a "tight" upper and lower-bound we use the Theta notation
  - $T(n) = \Theta(f(n))$ if $T(n)=O(f(n))$ and $f(n)=O(T(n))$
  - e.g., $3n^2-n = \Theta(n^2)$
- If $T(n) = \Theta(f(n))$ and $R(n) = \Theta(f(n))$, $T(n) + R(n) = \Theta(f(n))$

$\simeq$ and $\ll$

- Asymptotically equal: $f(n) \simeq g(n)$ if $\lim_{n\to\infty} f(n)/g(n) = 1$
  - i.e., eventually, f(n) and g(n) are equal (up to lower order terms)
  - If $\exists c>0$ s.t. $f(n) \simeq c \cdot g(n)$ then $f(n) = \Theta(g(n))$
    (for f(n) and g(n) which are eventually positive)
- Asymptotically much smaller: $f(n) \ll g(n)$ if $\lim_{n\to\infty} f(n)/g(n) = 0$
  - If $f(n) \ll g(n)$ then $f(n) = O(g(n))$ but $f(n) \neq \Theta(g(n))$
    (for f(n) and g(n) which are eventually positive)
- Note: Not necessary conditions: Θ and O do not require the limit to exist (e.g., $f(n) = n$ for odd n and $2n$ for even n: then $f(n) = \Theta(n)$)