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:
  - $T(0) = 0$ & $T(n) = T(n-1) + n^2 \quad \forall n \geq 1$
    - $T(n-1) = T(n-2) + (n-1)^2$, $T(n-2) = T(n-3) + (n-2)^2$, ...
    - $T(n) = n^2 + (n-1)^2 + (n-2)^2 + T(n-3) \quad \forall n \geq 3$
    - $T(n) = \Sigma_{k=1 \text{ to } n} \; k^2 + T(0) \qquad \forall n \geq 0$

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?

A slowly growing function

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

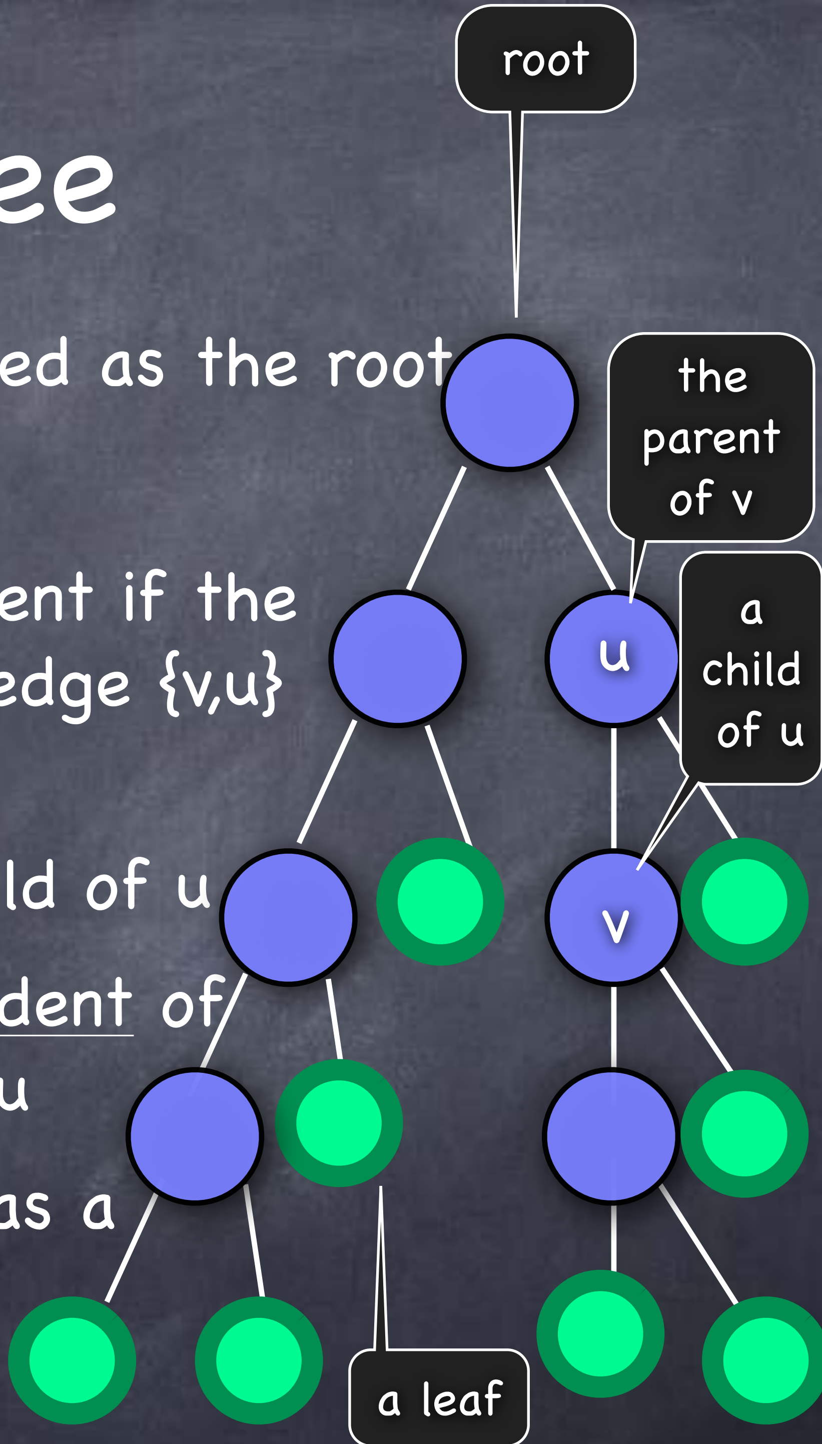
Tower of Hanoi

- 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”

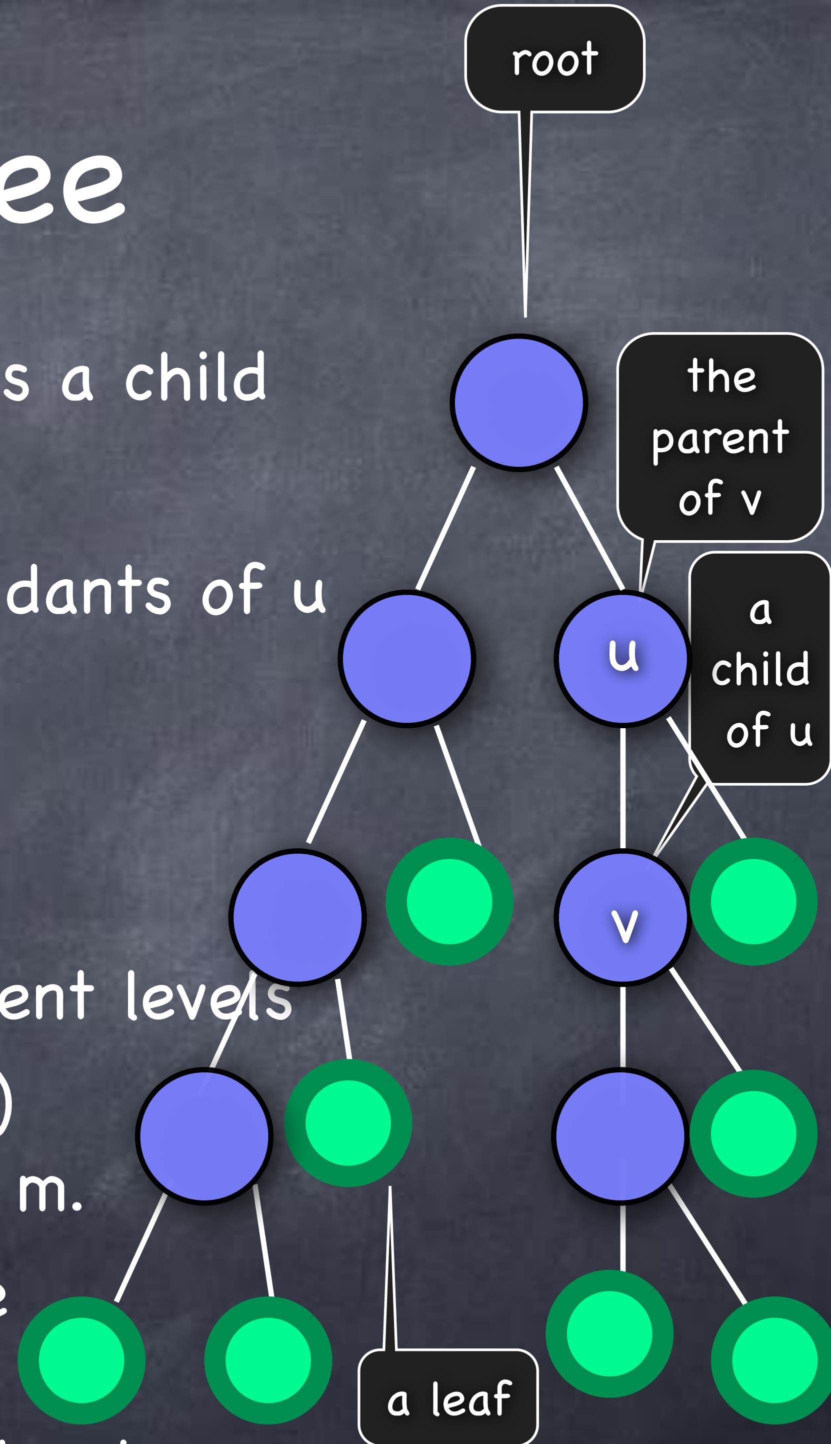
Rooted Tree

- 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)



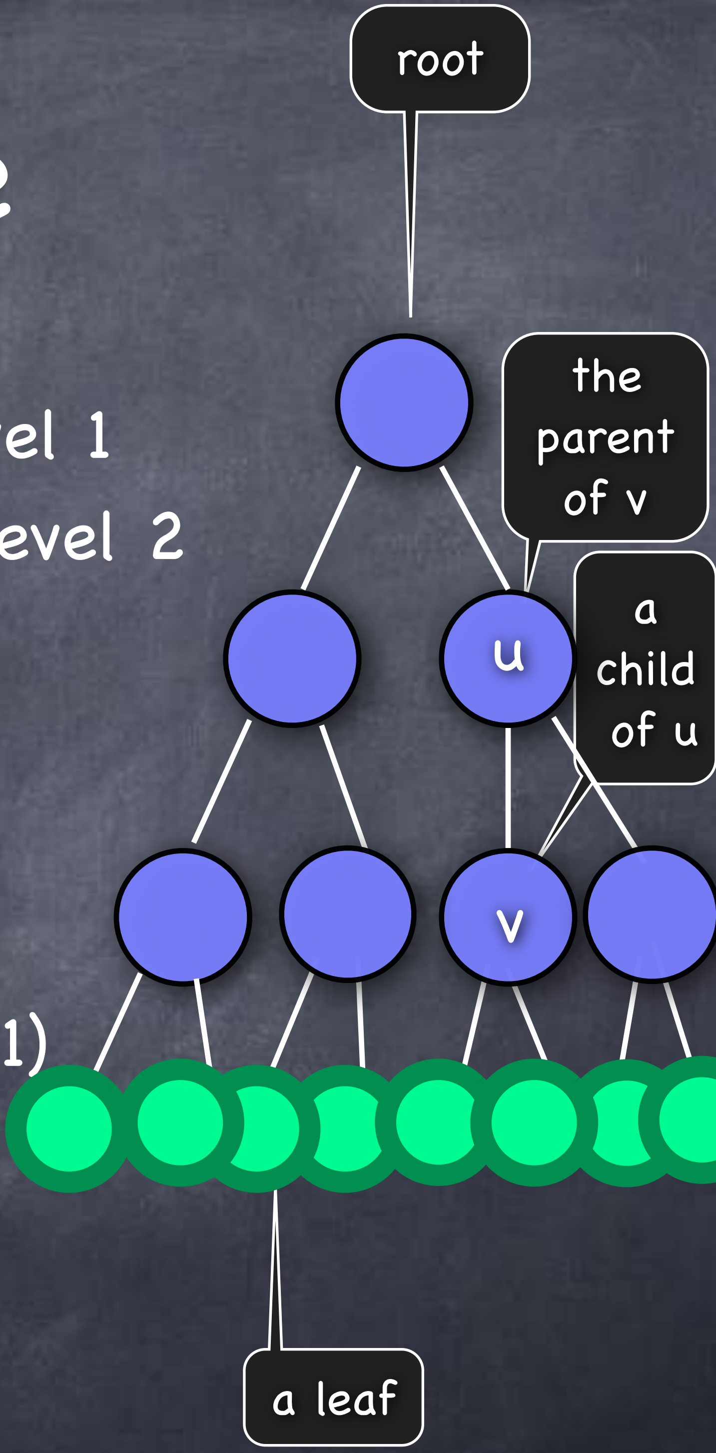
Rooted Tree

- 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 ≤ m.
- Full m-ary tree: Every internal node has exactly m children.
  Complete & Full: All leaves at same level



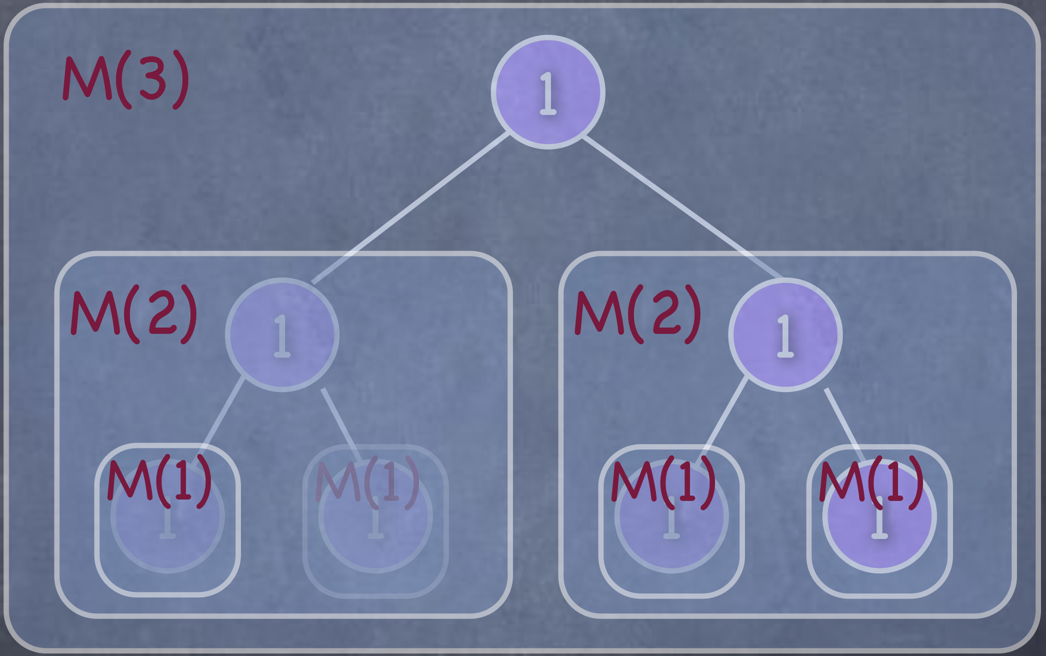
Rooted Tree

- 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:
  - $m^0 + m^1 + m^2 + \dots + m^h = (m^{h+1}-1)/(m-1)$
    - 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
- $M(n) = 2^n - 1$