DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

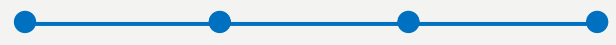
RECURSION TREE

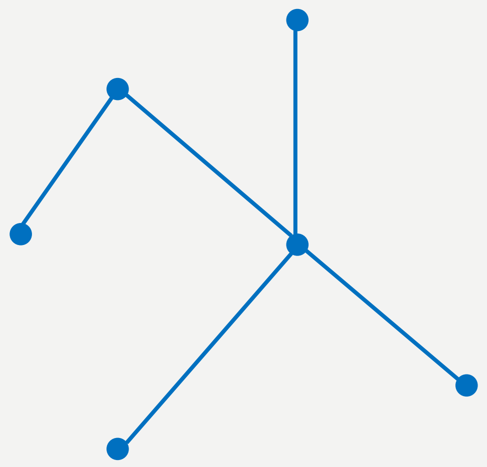
AGENDA

- Tree
- Recursion tree
  – Best for **making a good guess**

CONCEPTS OF TREES

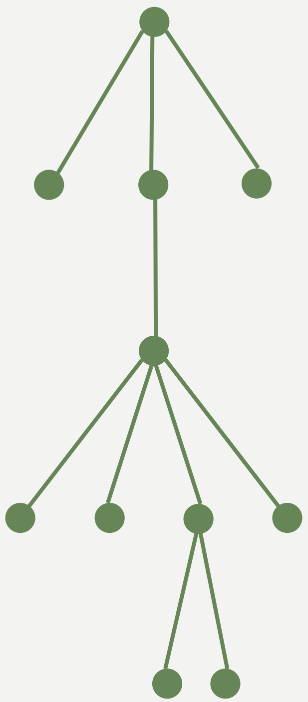
- A **tree** (T) is a connected ***acyclic undirected*** graph
  - In CS, some tress might be directed
- ***Unique path*** between two vertices
- A tree with n vertices has exactly $\boldsymbol{n-1}$ **edges**





TREE, FREE TREE, ROOTED TREE (V, E)

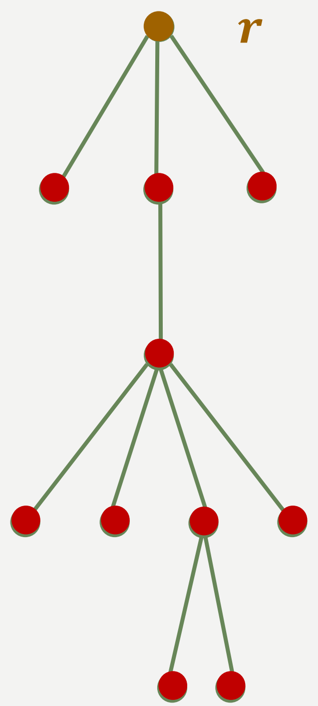
- A **tree**, $\boldsymbol{T}$, is often denoted by (V, E)
  - V is the set of vertices vertices
  - E is the set of edges
- A ***free tree*** is a **connected**, **acyclic**, **undirected** graph.
  - We often omit the adjective "~~***free***~~" when we say that a graph is a tree.
- A ***rooted* tree** is a *~~free~~ tree* in which one of the vertices is distinguished from the others.



ROOTED TREE (V, E) THE ROOT

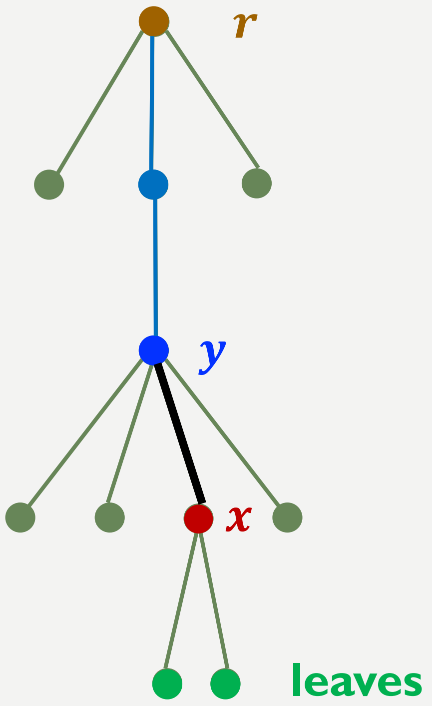
- The <u>distinguished</u> vertex of the ***rooted tree*** is called the ***root***.
  - Denoted by r.

- A vertex of a ***rooted tree*** is referred to as a ***node*** of the tree.



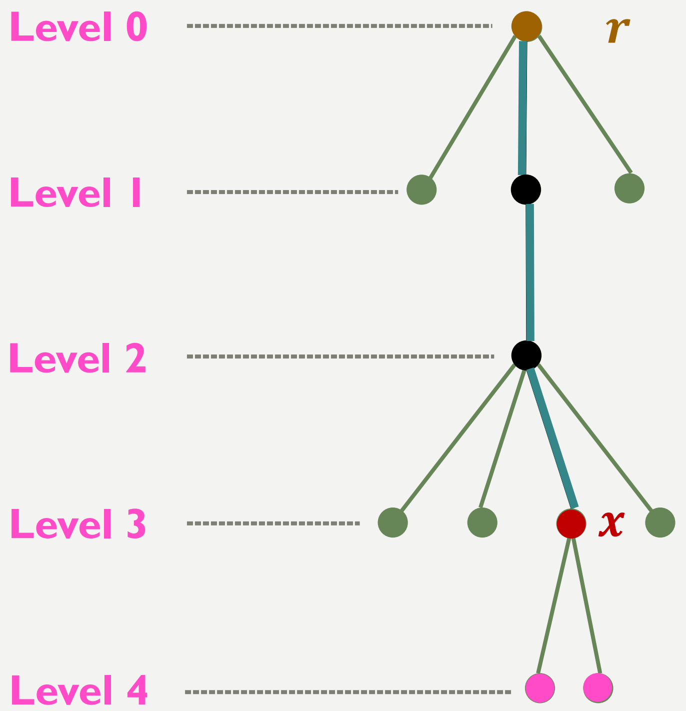
ROOTED TREE (V, E)
PARENT AND CHILDREN

- If the last edge on the simple path from the root r of a **tree** T to a node x is (y, x), then y is the ***parent*** of x, and x is a ***child*** of y.
  - The ***root*** is the only node **without** a parent.
  - A node with no children is a ***leaf*** or ***external node***.
  - A nonleaf node is an **internal node**.



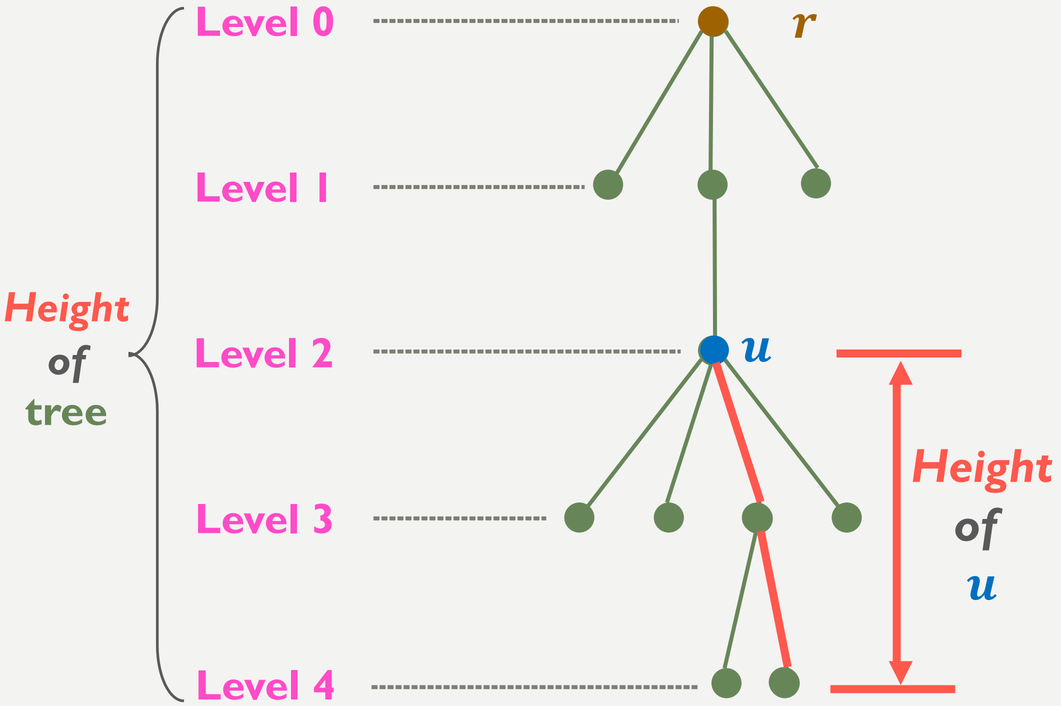
ROOTED TREE (V, E) DEGREE, DEPTH, LEVEL

- The number of children of a node x in a ***rooted tree*** T equals the **degree** of x.
- The length of the simple path from the ***root*** r to a node x is the ***depth*** of x in T.
  - A *level* of a tree consists of all nodes at the same ***depth***.



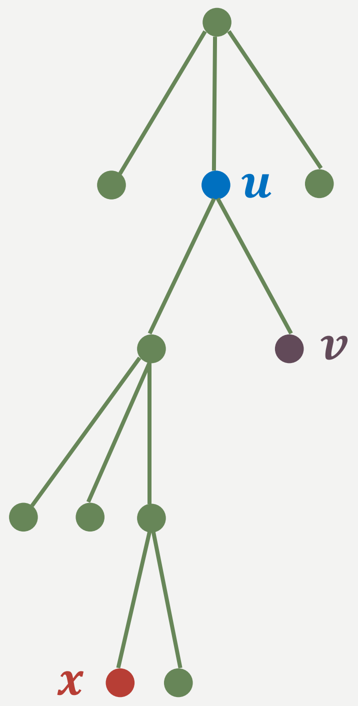
ROOTED TREE (V, E)
HEIGHT

- The ***height*** of a node in a tree is **the number of edges** on the longest simple downward path from the node to a leaf.
  - The height of u is **2**.
  - The ***height*** of a tree is the ***height*** of its ***root*** r.



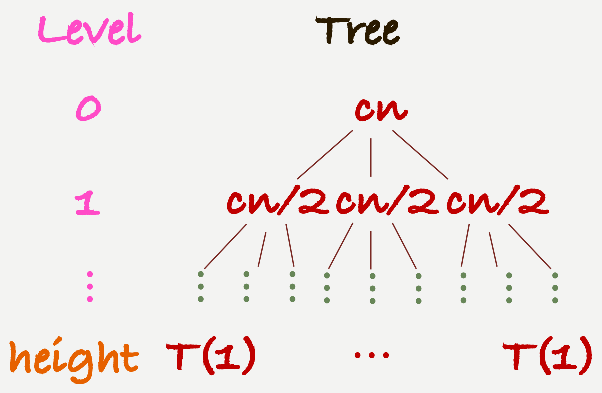
ROOTED TREE PRACTICE

- Given the tree $T = (V, E)$ on the right.
  - $|V| =$ __11__, $|E| =$ __10__.
  - The degree of $\boldsymbol{u}$ is __2__; the degree of $\boldsymbol{x}$ is __0__.
  - Node $\boldsymbol{v}$ is at level __2__; the root is at level __0__.
  - The height of $\boldsymbol{u}$ is __3__; the height of $\boldsymbol{v}$ is __0__.
  - The height of the tree is __4__.



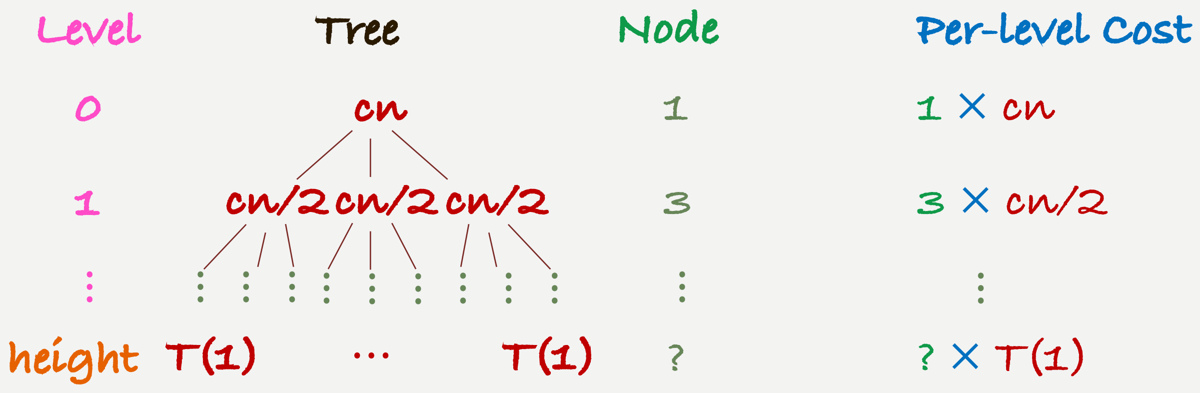
RECURSION TREE
A NODE

- **Each node** represents **the cost of a single subproblem** somewhere in the set of recursive function invocations.



RECURSION TREE
PER-LEVEL COSTS

- A set of ***per-level costs*** $\boldsymbol{f(k)}$ can be obtained by summing the costs within each level of the tree.
  - We often draw a **NODE** column to store the number of nodes at a level.

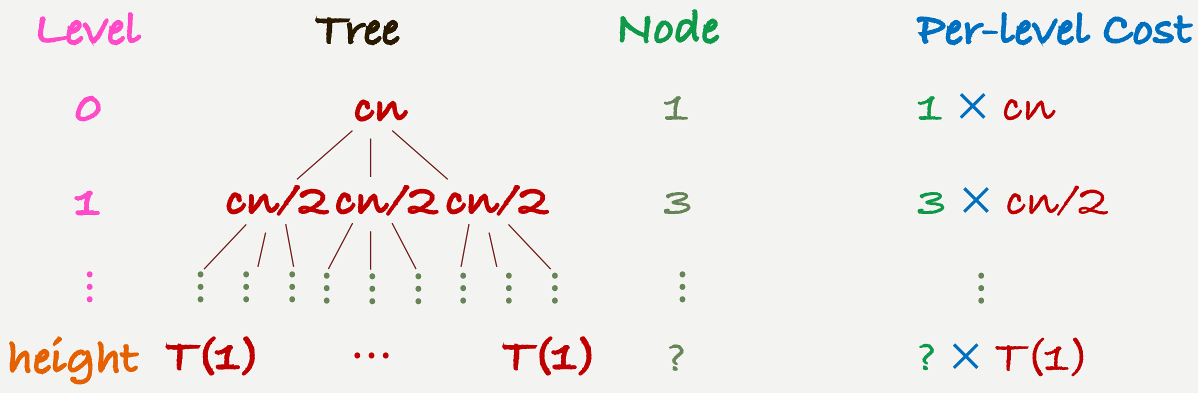


RECURSION TREE
TOTAL COST

- **Sum** all the ***per-level costs*** to determine the total cost of all levels of the recursion.

$$T(n) = \sum_{k=0}^{height} f(k)$$

  - $height = ?$



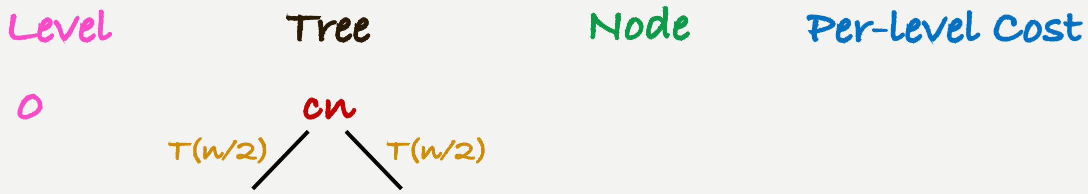
RECURSION TREE IN ACTION
STEP 1

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\ , & n > 1 \end{cases}$ by using recursion tree.
- **Step 1**: Draw the "head" of the tree.
  - **Level**
  - Tree (the recursion tree)
  - **Node** (# of **nodes** at a level)
  - **Per-level cost** (cost within a level)

Level	Tree	Node	Per-level Cost

RECURSION TREE IN ACTION
STEP 2 (LEVEL 0)

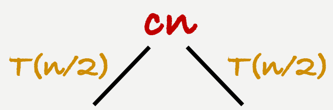
- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\,, & n > 1 \end{cases}$ by using recursion tree.
- **Step 2**: Start at **level 0**, draw the tree **downto level 2**.
  - The cost of **level 0** $\boldsymbol{T(n)}$ consists of
    - $\Theta(n) = \boldsymbol{cn}$
    - $T(n/2)$
    - $T(n/2)$



RECURSION TREE IN ACTION
STEP 2 (LEVEL 0)

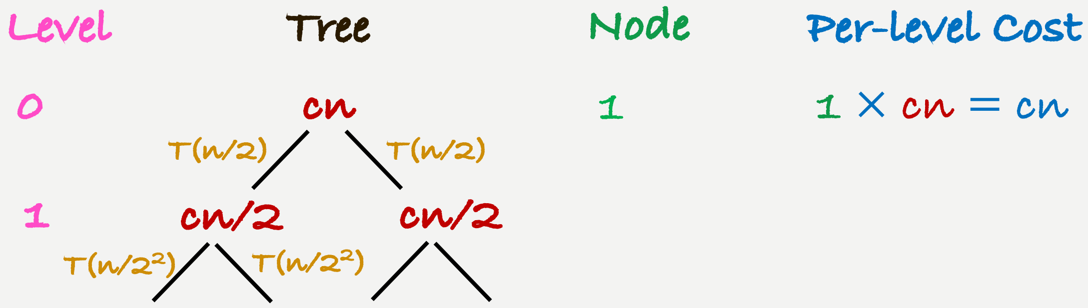
- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\,, & n > 1 \end{cases}$ by using recursion tree.
- **Step 2**: Start at **level 0**, draw the tree **downto level 2**.
  - Complete
    - **Node** (# of **nodes** at **level 0**)
    - **Per-level cost** (cost within **level 0**)

Level	Tree	Node	Per-level Cost
0	cn	1	$1 \times cn = cn$



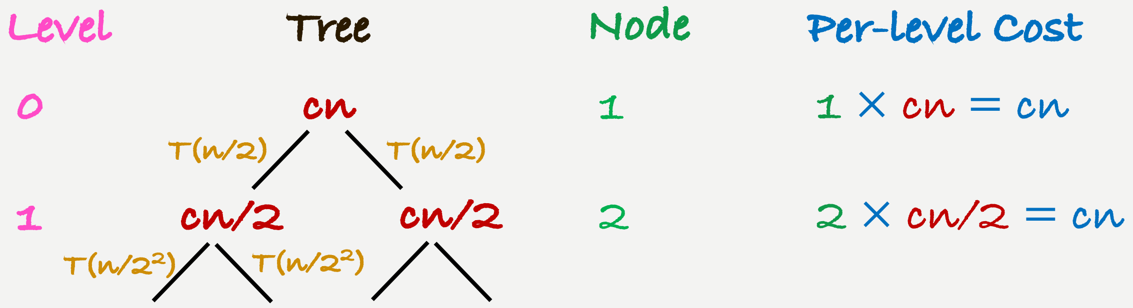
RECURSION TREE IN ACTION
STEP 2 (LEVEL 1)

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\ , & n > 1 \end{cases}$ by using recursion tree.
- **Step 2**: Start at **level 0**, draw the tree **downto** **level 2**.
  - **Each** node of **level 1** $\boldsymbol{T(n/2)}$ consists of
    - $\Theta(n/2) = \boldsymbol{cn/2}$
    - $T(n/2^2)$
    - $T(n/2^2)$



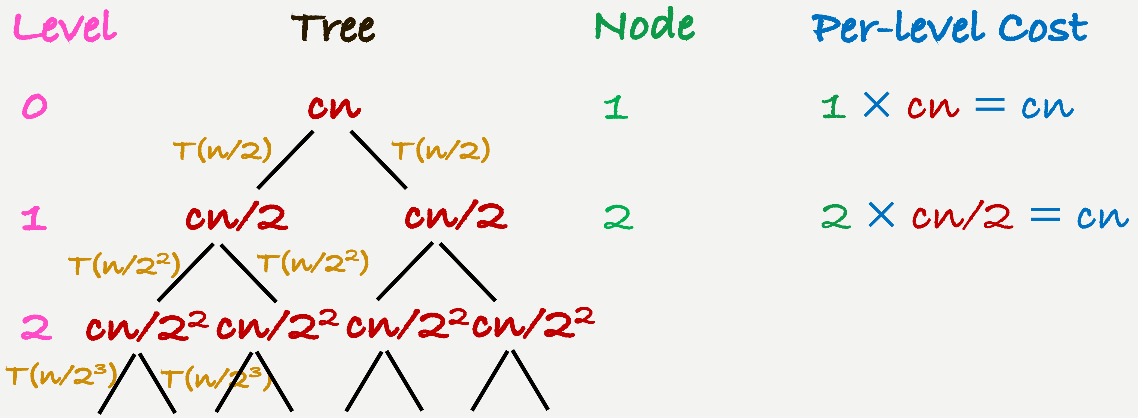
RECURSION TREE IN ACTION
STEP 2 (LEVEL 1)

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\ , & n > 1 \end{cases}$ by using recursion tree.
- **Step 2**: Start at **level 0**, draw the tree **downto level 2**.
  - Complete
    - **Node** (# of **nodes** at **level 1**)
    - **Per-level cost** (cost within **level 1**)



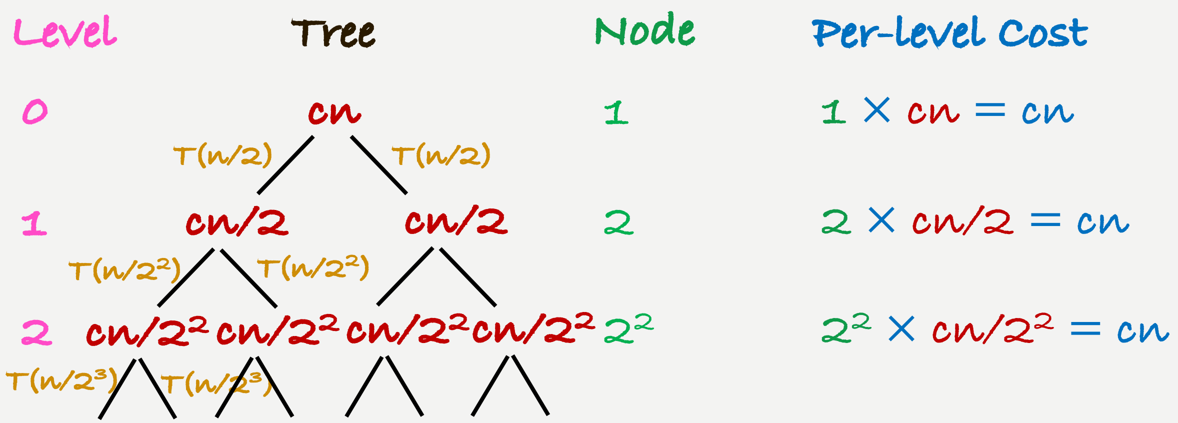
RECURSION TREE IN ACTION
STEP 2 (LEVEL 2)

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\ , & n > 1 \end{cases}$ by using recursion tree.
- **Step 2**: Start at **level 0**, draw the tree **downto** **level 2**.
  - **Each** node **level 2** $\boldsymbol{T(n/2^2)}$consists of
    - $\Theta(n/2^2) = \boldsymbol{cn/2^2}$
    - $T(n/2^3)$
    - $T(n/2^3)$



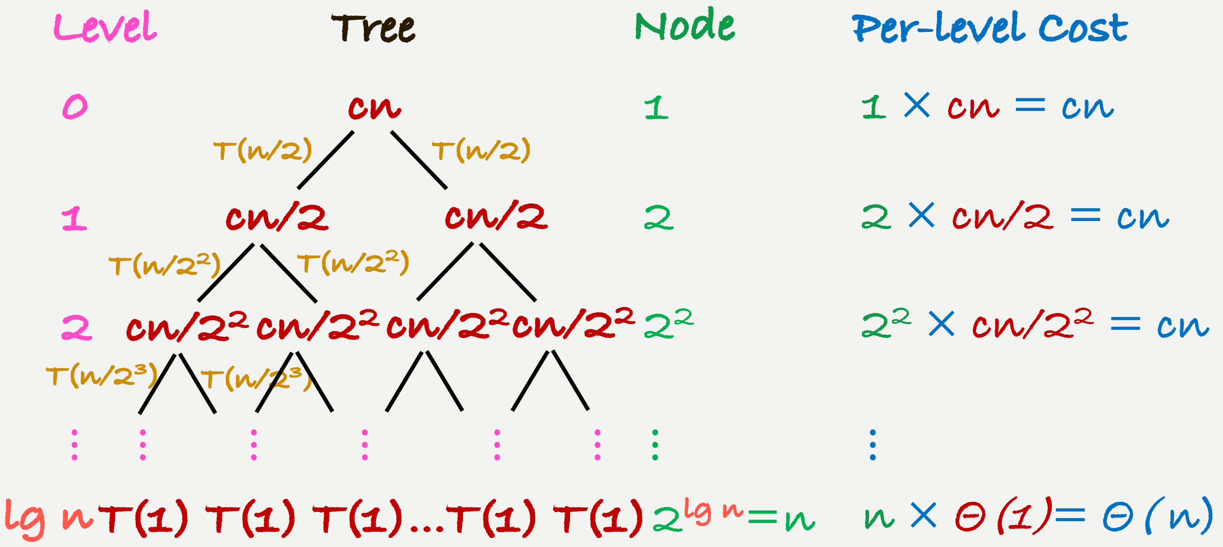
RECURSION TREE IN ACTION
STEP 2 (LEVEL 2)

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\ , & n > 1 \end{cases}$ by using recursion tree.
- **Step 2**: Start at **level 0**, draw the tree **downto** **level 2**.
  - Complete
    - **Node** (# of **nodes** at **level 2**)
    - **Per-level cost** (cost within **level 2**)



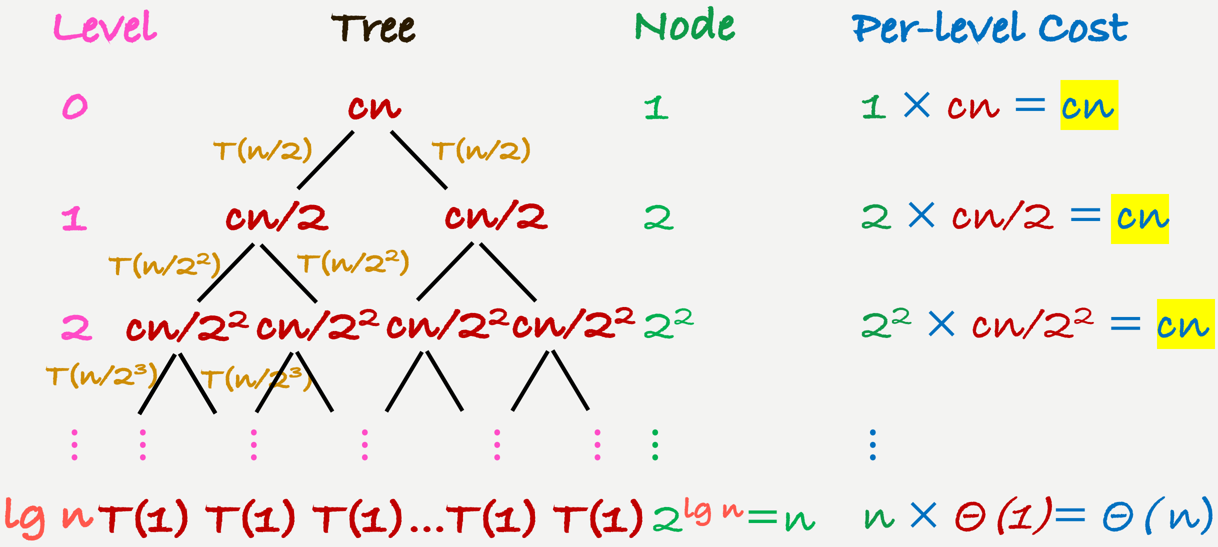
RECURSION TREE IN ACTION
STEP 3

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n), & n > 1 \end{cases}$ by using recursion tree.
- **Step 3**: Complete the recursion tree by drawing the deepest **level**.
  - *height* $= \mathbf{lg}\ \boldsymbol{n}$
    - Assume without loss of generality that n is a power of 2.



RECURSION TREE IN ACTION
STEP 4

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)\ , & n > 1 \end{cases}$ by using recursion tree.
- **Step 4**: Derive the cost of **level k** as a function of $\boldsymbol{k}$.
  $\boldsymbol{f(k) = \ cn}$

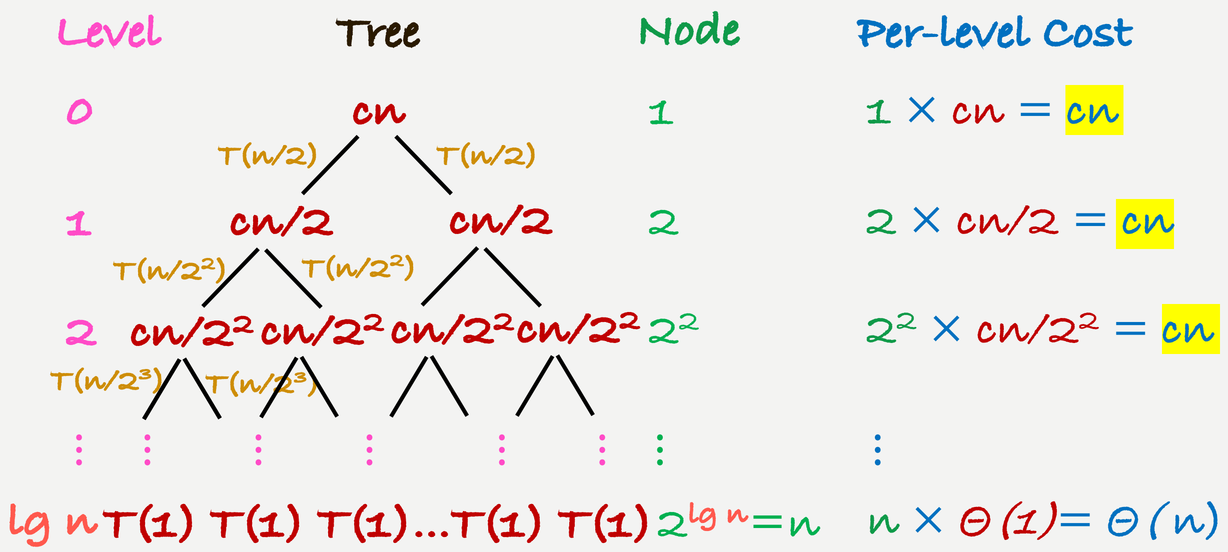


RECURSION TREE IN ACTION
STEP 5

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n), & n > 1 \end{cases}$ by using recursion tree.
- **Step 5**: Compute the **total cost** by summing all the **per-level costs** $\boldsymbol{f(k)}$ of all levels of the recursion.

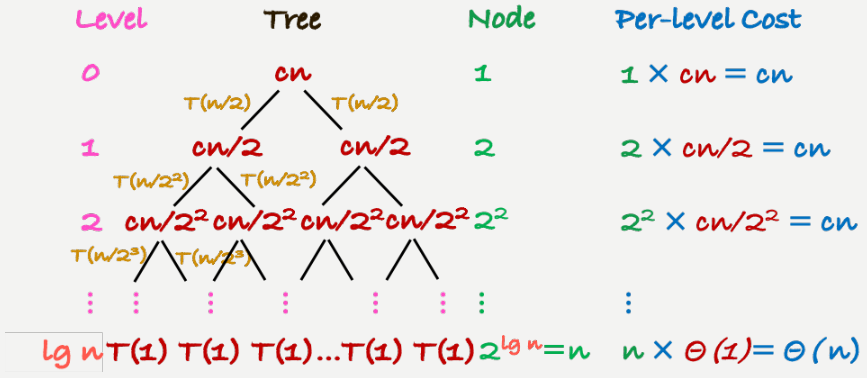
$$T(n) = \sum_{k=0}^{\lg n} cn$$

$$= cn(\lg n + 1) = \Theta(n \lg n)$$



RECURSION TREE IN ACTION
REVIEW

- **Step 1**: Draw the "head" of the tree.
- **Step 2**: Start at **level 0**, draw the tree **downto** **level 2**.
- **Step 3**: Complete the recursion tree by drawing the deepest **level**.



- **Step 4**: Derive the cost of **level** $\boldsymbol{k}$ as a function of $\boldsymbol{k}$.
- **Step 5**: Compute the **total cost** by summing all the **per-level costs** $\boldsymbol{f(k)}$ of all levels of the recursion.

$$T(n) = \sum_{\boldsymbol{k=0}}^{\boldsymbol{height}} \boldsymbol{f(k)}$$

RECURSION TREE IN ACTION
PRACTICE

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 3 \cdot T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2), & n > 1 \end{cases}$ by using recursion tree.
- Tolerate sloppiness $\Longrightarrow T(n) = 3 \cdot T\left(\frac{n}{4}\right) + cn^2$
- **Step 1 through step 3**

RECURSION TREE IN ACTION
PRACTICE

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 3 \cdot T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2), & n > 1 \end{cases}$ by using recursion tree.
- **Step 4**:
  $f(k) =$

RECURSION TREE IN ACTION
PRACTICE

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 3 \cdot T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2), & n > 1 \end{cases}$ by using recursion tree.

- **Step 5**:

$$\boldsymbol{T(n)} = \sum_{\boldsymbol{k=0}}^{\underline{\qquad\qquad}} (\underline{\qquad\qquad})$$

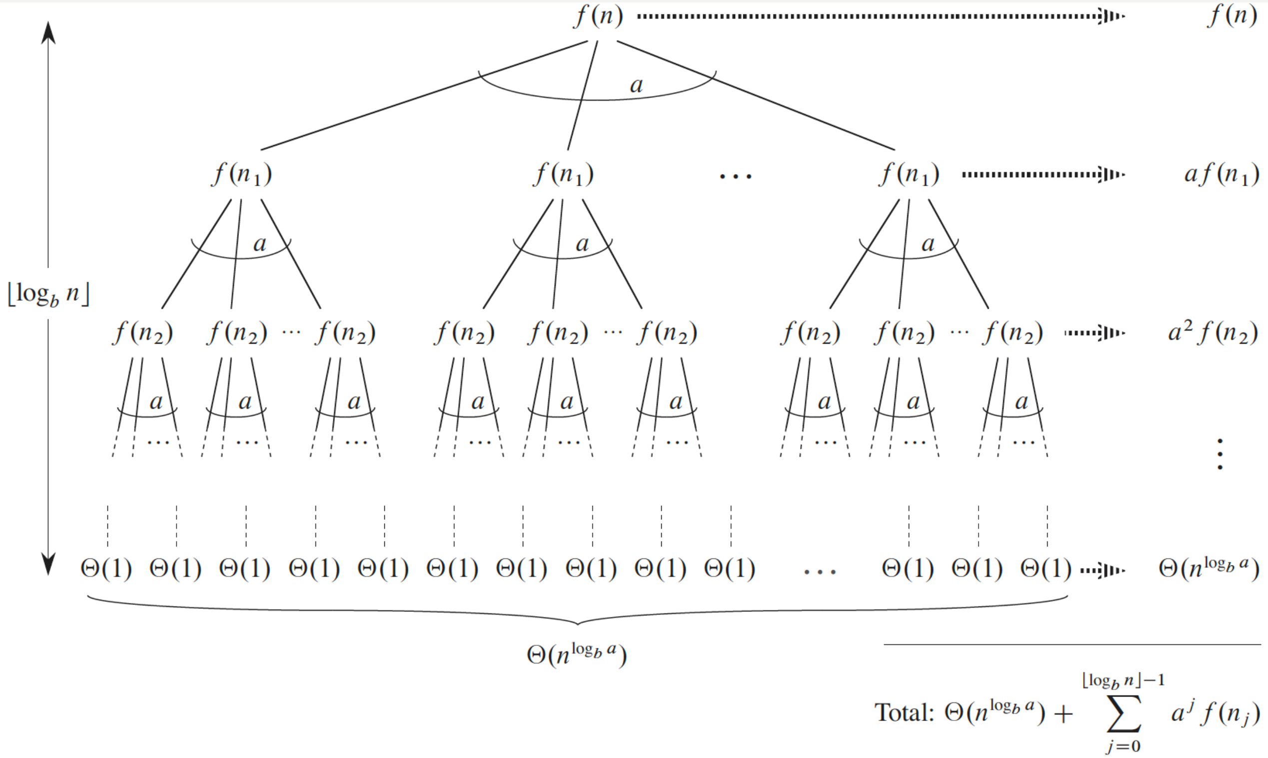
$$= \underline{\qquad\qquad\qquad\qquad}$$

$$= \boldsymbol{O}(\underline{\qquad\quad})$$

RECURSION TREE METHOD GENERALIZATION

- Consider recursive function in the form of $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$.
  - A problem is divided into a subproblems with each subproblem solving n/b of the input.
  - The cost of a single subproblem is a function $f(n_j)$, where j is the level number. $n_j = n/b^j$.

$$T(n) = \Theta\left(n^{\log_b a}\right) + \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)$$



RECURSION TREE METHOD
GENERALIZATION PRACTICE

- Make a guess of $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 3 \cdot T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2), & n > 1 \end{cases}$ by using recursion tree.

- **Solution (**Tolerate sloppiness $\Longrightarrow T(n) = 3 \cdot T\left(\frac{n}{4}\right) + cn^2$**)**
  - The recurrence must be in the form of $\boldsymbol{T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)}$. $\boldsymbol{a}$ = $\underline{\ \mathbf{3}\ }$, $\boldsymbol{b}$ = $\underline{\ \mathbf{4}\ }$, $\boldsymbol{f(n)}$ = $\underline{\boldsymbol{cn^2}}$.
  - Plug $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{f(n)}$ in the formula.

$$\boldsymbol{T(n) = \Theta\left(n^{\log_4 3}\right) + \sum_{j=0}^{\lfloor \log_4 n \rfloor - 1} 3^j\, cn^2} \approx \Theta\left(n^{\log_4 3}\right) + cn^2 \cdot \sum_{j=0}^{\log_4 n - 1} 3^j$$

$$< \Theta\left(n^{\log_4 3}\right) + \frac{1}{1 - (3/16)} cn^2 \quad = \boldsymbol{O(n^2)}$$

NEXT UP
MASTER THEOREM

- Solving recurrence

REFERENCE