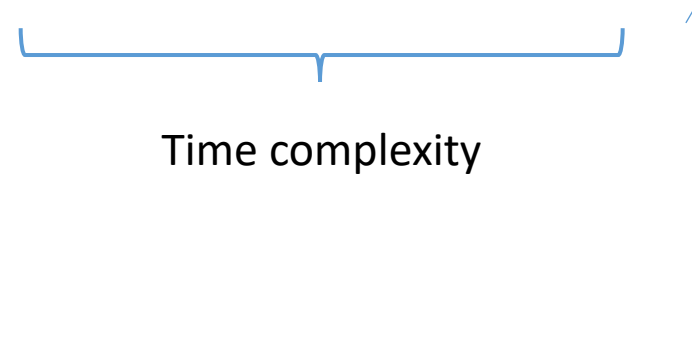


CS3230 – Design and Analysis of Algorithms
(S1 AY2025/26)

Lecture 2: Recurrences and Master Theorem

Analyzing the running time of an algorithm

- **Goal:** For a given algorithm \mathcal{A} , analyze the asymptotic running time $T(n)$ as a function of the input size n .



Unless otherwise stated, we consider the worst-case running time.

- $T(n)$ is the worst-case running time over all possible inputs of size n .

Analyzing the running time of an algorithm

- **Goal:** For a given *recursive* algorithm \mathcal{A} , analyze the asymptotic running time $T(n)$ as a function of the input size n .
 - Step 1: Derive a recurrence.
 - Step 2: Solve the recurrence.

Analyzing the running time of an algorithm

recursive

- **Goal:** For a given algorithm \mathcal{A} , analyze the asymptotic running time $T(n)$ as a function of the input size n .
 - Step 1: Derive a recurrence.
 - Step 2: Solve the recurrence.

Fib(n)

- If $n \leq 1$, return n .
- Else, return **Fib**($n - 1$) + **Fib**($n - 2$).

Step 1



$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n-1) + T(n-2) + \Theta(1) & \text{if } n > 1 \end{cases}$$



Step 2

$$T(n) \in \Omega(2^{n/2})$$

Merge sort

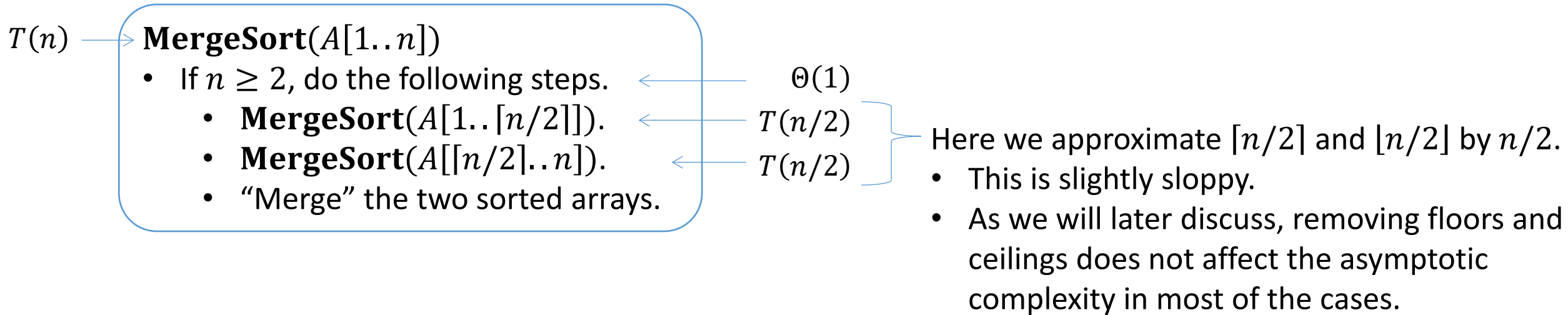
- $T(n)$ → **MergeSort**($A[1..n]$)
- If $n \geq 2$, do the following steps.
 - **MergeSort**($A[1.. \lfloor n/2 \rfloor]$).
 - **MergeSort**($A[\lceil n/2 \rceil..n]$).
 - “Merge” the two sorted arrays.



We omit the details.

Question: How to derive a recurrence for the running time $T(n)$ of Merge sort?

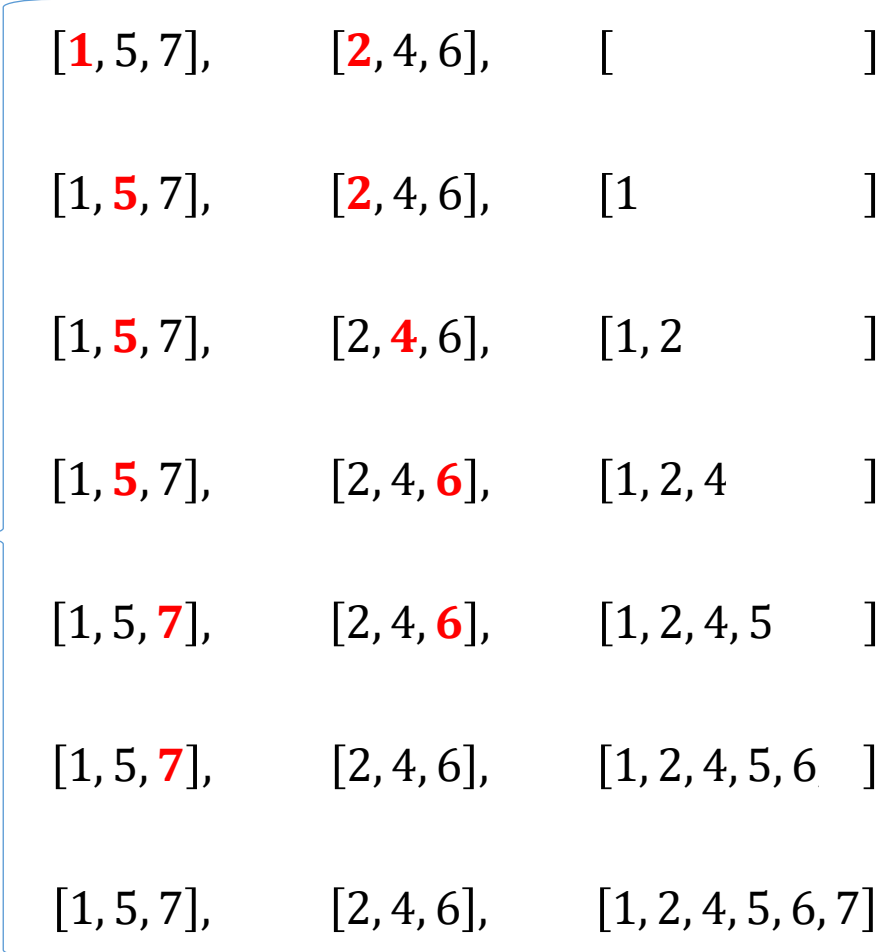
Merge sort



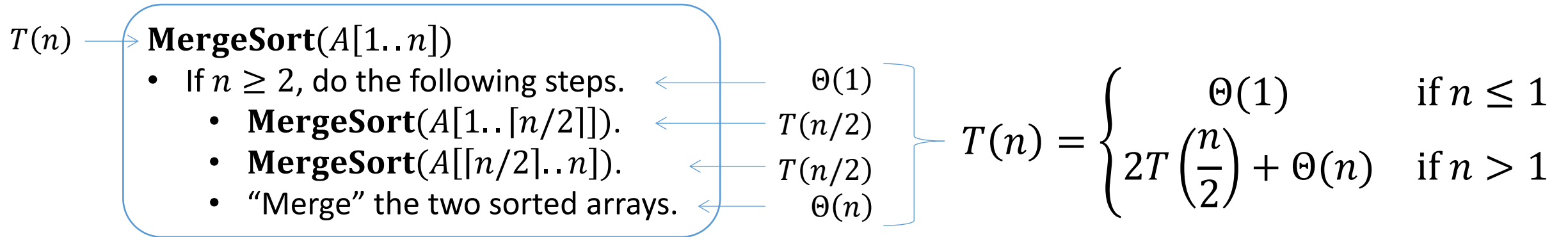
Merge sort

$T(n)$ → **MergeSort**($A[1..n]$)

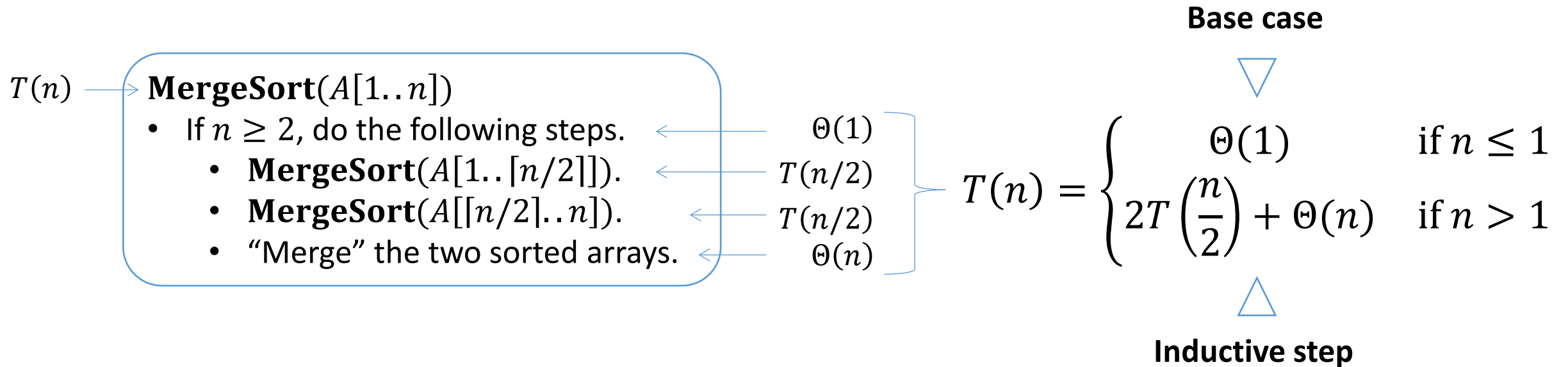
- If $n \geq 2$, do the following steps. ← $\Theta(1)$
 - **MergeSort**($A[1.. \lfloor n/2 \rfloor]$). ← $T(n/2)$
 - **MergeSort**($A[\lfloor n/2 \rfloor .. n]$). ← $T(n/2)$
 - “Merge” the two sorted arrays. ← $\Theta(n)$



Merge sort



Merge sort



Note: We often omit stating the base case because $T(n)$ is $\Theta(1)$ whenever $n \in O(1)$.

- The precise constant does not matter in most of the cases.

Solving a recurrence

- How to solve a given recurrence:
 - Merge sort: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$
- **Four methods:**
 - Telescoping
 - Substitution
 - Recursion tree
 - Master theorem

Solving a recurrence

- How to solve a given recurrence:
 - Merge sort: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Remark:

If $f(n) \in O(n)$, then there exist two constants $c > 0$ and $n_0 > 0$ such that $f(n) \leq cn$ if $n \geq n_0$.

- For upper bound calculation, we can replace $\Theta(n)$ with cn .
 - $T(n) \leq 2T\left(\frac{n}{2}\right) + cn$ (if $n \geq n_0$).

If $f(n) \in \Omega(n)$, then there exist two constants $c > 0$ and $n_0 > 0$ such that $f(n) \geq cn$ if $n \geq n_0$.

- For lower bound calculation, we can replace $\Theta(n)$ with cn .
 - $T(n) \geq 2T\left(\frac{n}{2}\right) + cn$ (if $n \geq n_0$).

Telescoping series

- **An example:**

$$\begin{aligned}\sum_{k=1}^n \frac{1}{k(k+1)} &= \sum_{k=1}^n \left(\frac{1}{k} - \frac{1}{k+1} \right) \\ &= \left(\frac{1}{1} - \frac{1}{2} \right) + \left(\frac{1}{2} - \frac{1}{3} \right) + \cdots + \left(\frac{1}{n-1} - \frac{1}{n} \right) + \left(\frac{1}{n} - \frac{1}{n+1} \right) \\ &= 1 - \frac{1}{n+1}\end{aligned}$$

Telescoping method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1 \end{cases}$



For the sake of simplicity, we omit $\Theta(\cdot)$ here and assume that $n = 2^k$ for some integer $k \geq 0$.

Telescoping method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1 \end{cases}$

$\frac{T(n)}{n} = \frac{T\left(\frac{n}{2}\right)}{\frac{n}{2}} + 1$

$\frac{T\left(\frac{n}{2}\right)}{\frac{n}{2}} = \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}} + 1$

$\frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}} = \frac{T\left(\frac{n}{8}\right)}{\frac{n}{8}} + 1$

\vdots

$\frac{T(2)}{2} = \frac{T(1)}{1} + 1$

$\log n$

Telescoping method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1 \end{cases}$

$$\frac{T(n)}{n} = \frac{T(1)}{1} + \log n$$

$$T(n) \in \Theta(n \log n)$$

$$\frac{T(n)}{n} = \frac{T\left(\frac{n}{2}\right)}{\frac{n}{2}} + 1$$

$$\frac{T\left(\frac{n}{2}\right)}{\frac{n}{2}} = \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}} + 1$$

$$\frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}} = \frac{T\left(\frac{n}{8}\right)}{\frac{n}{8}} + 1$$

⋮

$$\frac{T(2)}{2} = \frac{T(1)}{1} + 1$$

log n

Telescoping method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 4T\left(\frac{n}{2}\right) + n & \text{if } n > 1 \end{cases}$

$$\frac{T(n)}{n^2} = \frac{T(1)}{1^2} + \left(\frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{n}\right)$$

$$T(n) \in \Theta(n^2)$$

$\log n$

$$\frac{T(n)}{n^2} = \frac{T\left(\frac{n}{2}\right)}{\left(\frac{n}{2}\right)^2} + \frac{1}{n}$$

$$\frac{T\left(\frac{n}{2}\right)}{\left(\frac{n}{2}\right)^2} = \frac{T\left(\frac{n}{4}\right)}{\left(\frac{n}{4}\right)^2} + \frac{2}{n}$$

$$\frac{T\left(\frac{n}{4}\right)}{\left(\frac{n}{4}\right)^2} = \frac{T\left(\frac{n}{8}\right)}{\left(\frac{n}{8}\right)^2} + \frac{4}{n}$$

\vdots

$$\frac{T(2)}{2^2} = \frac{T(1)}{1^2} + \frac{1}{2}$$

Substitution method

- **Step 1:** Guess a solution.
- **Step 2:** Verify your solution by induction.

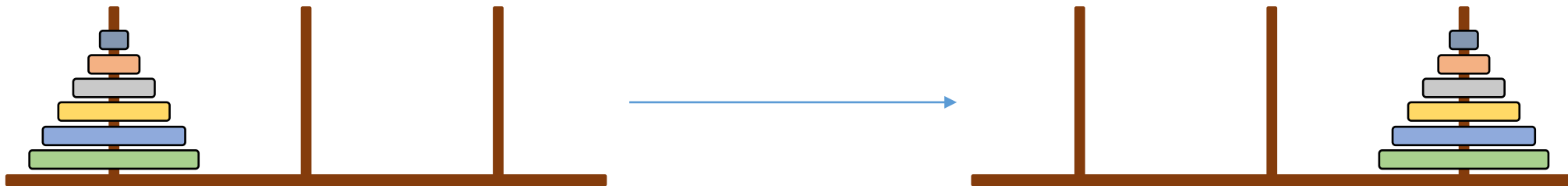
Tower of Hanoi

Rules:

- Only one disk may be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
- No disk may be placed on top of a disk that is smaller than it.

https://en.wikipedia.org/wiki/Tower_of_Hanoi

Goal:

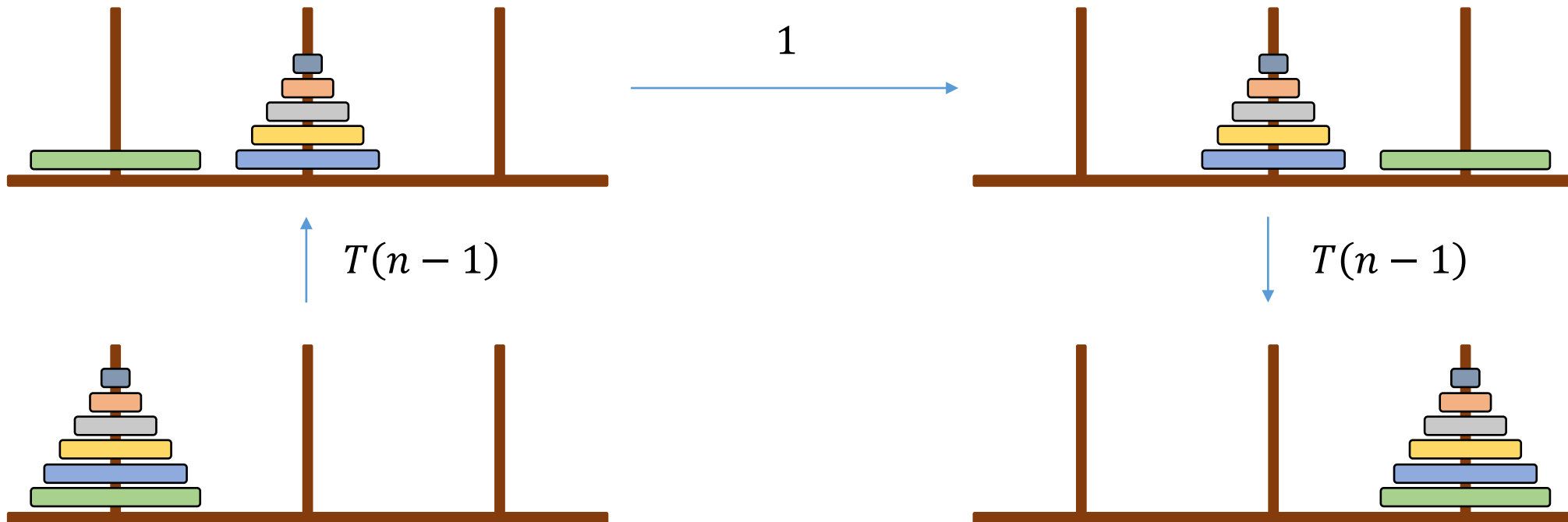


Tower of Hanoi

$T(n)$ = number of steps to move a stack of n disks.

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 2T(n - 1) + 1 & \text{if } n \geq 1 \end{cases}$$

A recursive algorithm:



Tower of Hanoi

$T(n)$ = number of steps to move a stack of n disks.

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 2T(n-1) + 1 & \text{if } n \geq 1 \end{cases}$$

Solving $T(n)$ by the **substitution method**:

- Step 1: Guess a solution:
 - $T(n) = 2^n - 1$
- Step 2: Verify your solution by induction.
 - **Base case:** $T(0) = 0 = 2^0 - 1$.
 - **Inductive step:** $T(n) = 2T(n-1) + 1 = 2 \cdot (2^{n-1} - 1) + 1 = 2^n - 1$

Recursion tree

- **Goal:** Solve the recurrence $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$

Recursion tree

- **Goal:** Solve the recurrence $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$

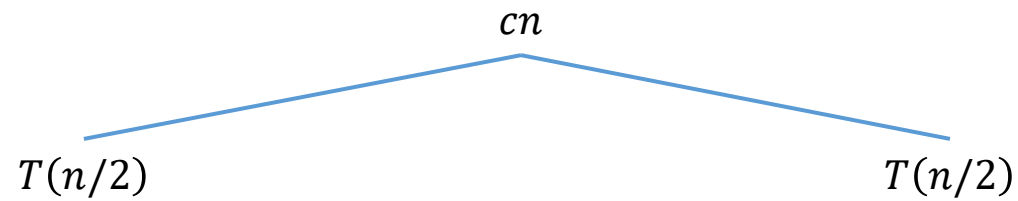
Use this custom code for a visualization of the recursion tree, with n being a power of two, e.g., 1, 2, 4, 8, 16, ...

```
if (n == 1) /* base case */  
    return 1;  
else /* recursive cases */  
    return f(n/2) + f(n/2);
```

Recursion tree

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$$

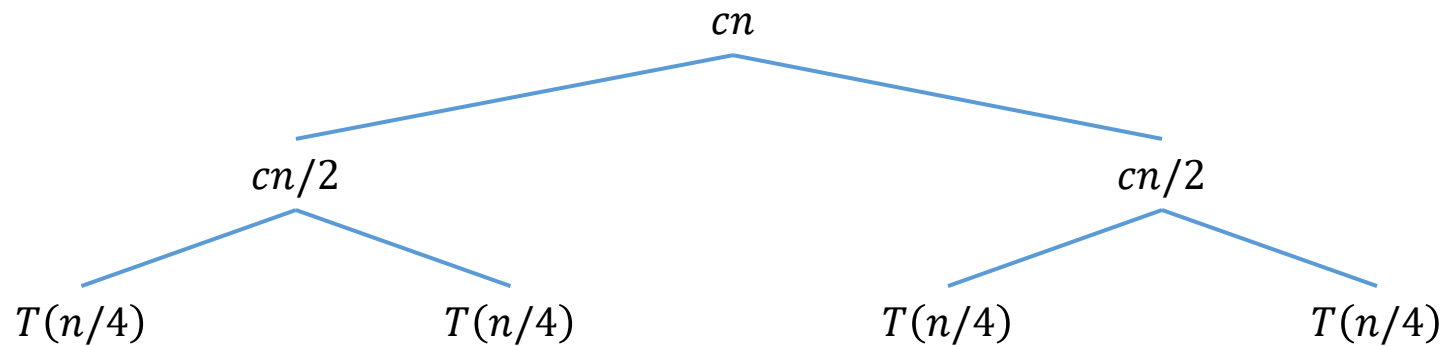
$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$



Recursion tree $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{cn}{2}$$

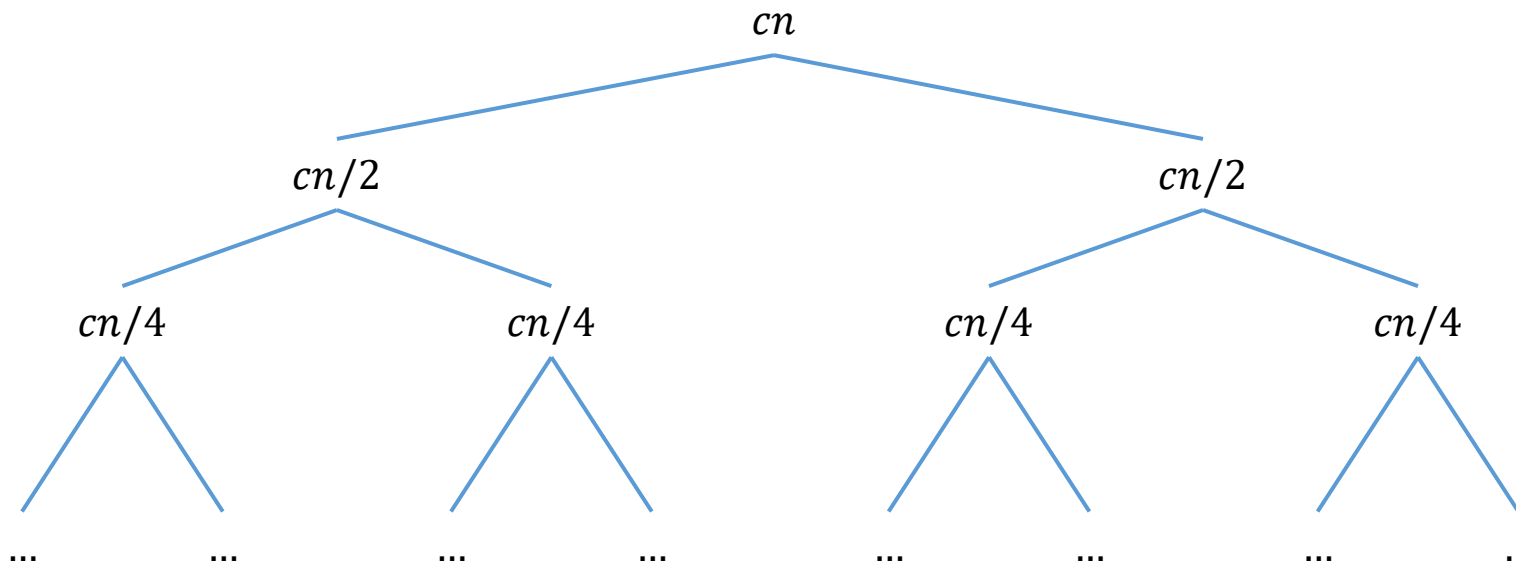


Recursion tree $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

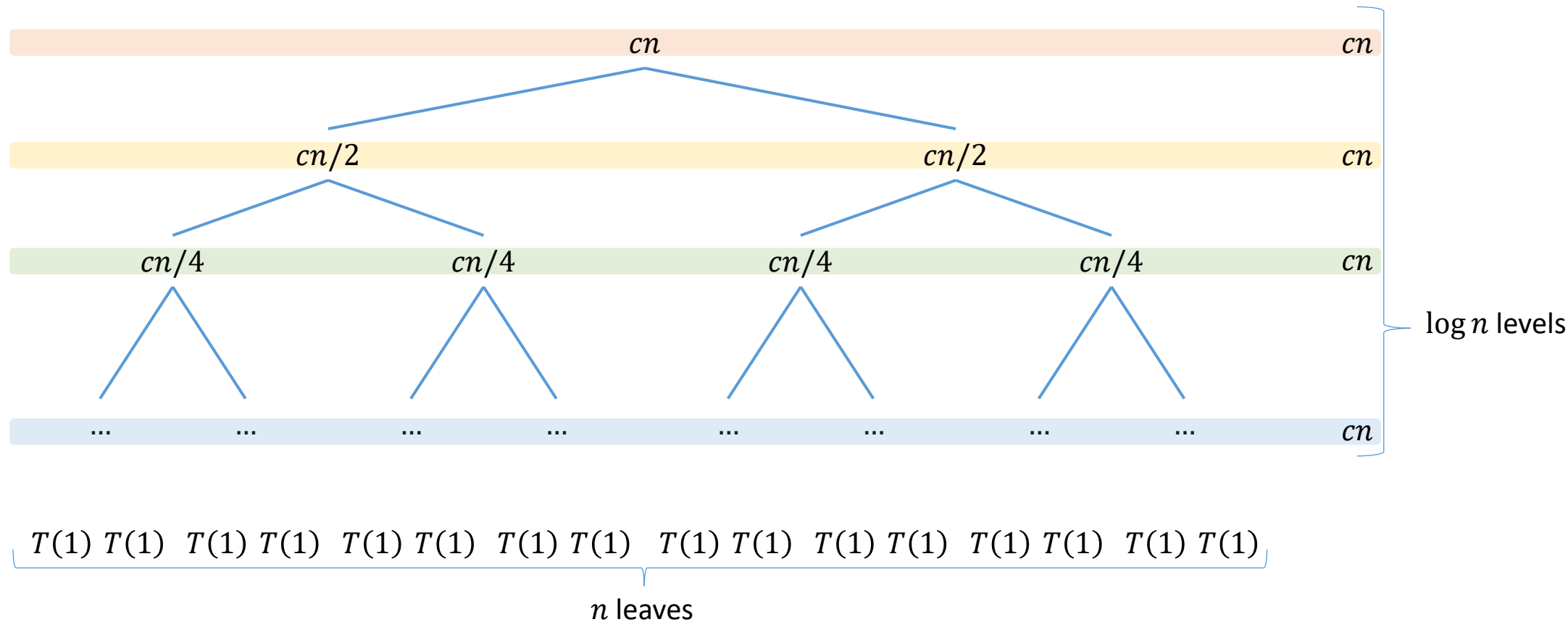
$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{cn}{2}$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{cn}{4}$$



$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$

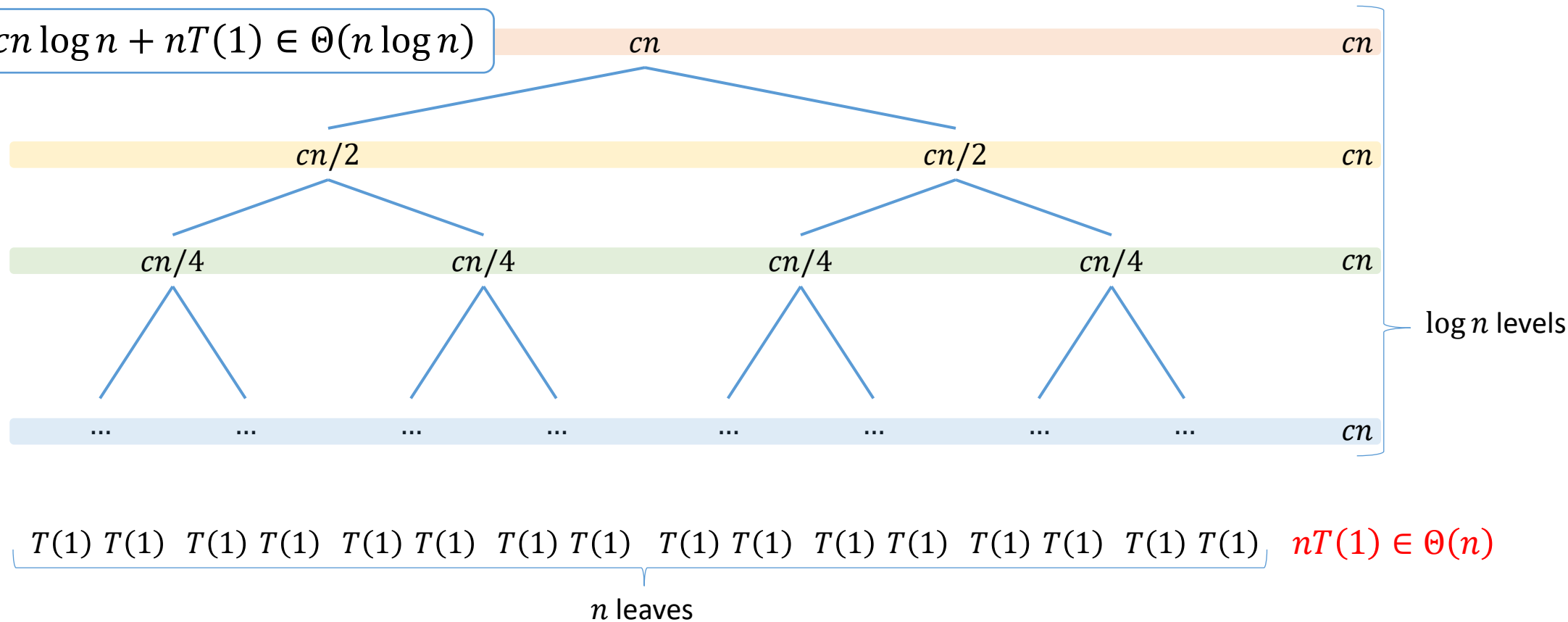
Recursion tree $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$



Recursion tree $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & \text{if } n > 1 \end{cases}$

$cn \log n \in \Theta(n \log n)$

$T(n) = cn \log n + nT(1) \in \Theta(n \log n)$



Question 1 @ VisuAlgo online quiz

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n-1) + T(1) + cn & \text{if } n > 1 \end{cases}$$

Which of the following statements is **true**?

- $T(n) \in \Theta(n)$
- $T(n) \in \Theta(n \log n)$
- $T(n) \in \Theta(n^2)$
- $T(n) \in \Theta(n^3)$

Question 2 @ VisuAlgo online quiz

Who is the **Master of Algorithms** pictured below?

- Robert Floyd
- Richard Karp
- Donald Knuth
- Alan Turing



Solving a recurrence of the generic form

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.
 - $a \geq 1$
 - $b \geq 1$
 - $f(n) \in \Omega(1)$
- **Goal:** Solve $T(n)$.

Solving a recurrence of the generic form

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.
 - $a \geq 1$
 - $b \geq 1$
 - $f(n) \in \Omega(1)$
- **Goal:** Solve $T(n)$.
- **Main idea:** Classify the work into two types and compare their costs.
 - Splitting/combining.
 - Solving the base cases.

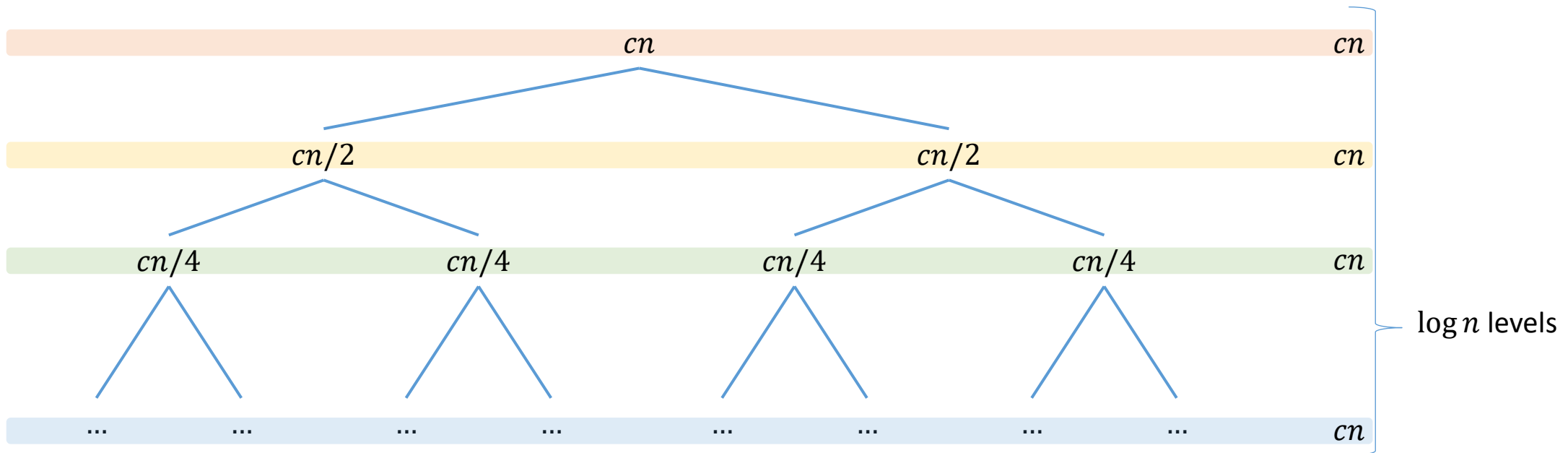
Two types of work

$$T(n) = cn \log n + nT(1) \in \Theta(n \log n)$$

Splitting/combining:

- Split a problem into sub-problems.
- Combine the solutions of subproblems.

$$cn \log n \in \Theta(n \log n)$$



$$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) \quad nT(1) \in \Theta(n)$$

n leaves

Solving the base cases:

- The cost is linear in the number of leaves.

Two types of work

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.



Solving the base cases

Splitting/combining



The number of leaves = n^d , where $d = \log_b a$ is the critical exponent.

Recursion tree:

- Tree height: $\log_b n$
- The number of children of a node: a
- The number of leaves: $a^{\log_b n} = n^{\log_b a} = n^d$

Master theorem

Two types of work:

- Solving the base cases: n^d
 - $d = \log_b a$
- Splitting/combining: $f(n)$

← Dominant term

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.

Case 1:

- $f(n) \in O(n^{d-\epsilon})$ for some constant $\epsilon > 0$.

▷ $T(n) \in \Theta(n^d)$

Master theorem

Two types of work:

- Solving the base cases: n^d
 - $d = \log_b a$
- Splitting/combining: $f(n)$

Comparable

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.

Case 1:

- $f(n) \in O(n^{d-\epsilon})$ for some constant $\epsilon > 0$.

▷ $T(n) \in \Theta(n^d)$

Case 2:

- $f(n) \in \Theta(n^d \log^k n)$ for some constant $k \geq 0$.

▷ $T(n) \in \Theta(n^d \log^{k+1} n)$

Master theorem

Two types of work:

- Solving the base cases: n^d
 - $d = \log_b a$
- Splitting/combining: $f(n)$

← Dominant term

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.

Case 1:

- $f(n) \in O(n^{d-\epsilon})$ for some constant $\epsilon > 0$.

▷ $T(n) \in \Theta(n^d)$

Case 2:

- $f(n) \in \Theta(n^d \log^k n)$ for some constant $k \geq 0$.

▷ $T(n) \in \Theta(n^d \log^{k+1} n)$

Case 3:

- $f(n) \in \Omega(n^{d+\epsilon})$ for some constant $\epsilon > 0$.
- $af(n/b) \leq cf(n)$ for some constant $c < 1$.

▷ $T(n) \in \Theta(f(n))$

A **regularity condition** ensuring that the splitting/combining cost $f(n)$ at the top level of recursion is the dominant term.

Examples

Two types of work:

- Solving the base cases: n^d
 - $d = \log_b a$
- Splitting/combining: $f(n)$

← Dominant term

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.

Critical exponent: $d = \log_b a$

Case 1:

- $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.

▷ $T(n) \in \Theta(n^d)$

Case 2:

- $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.

▷ $T(n) \in \Theta(n^d \log^{k+1} n)$

Case 3:

- $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
- $af(n/b) \leq cf(n)$ for some $c < 1$.

▷ $T(n) \in \Theta(f(n))$

Solve: $T(n) = 4T(n/2) + n$.

- $a = 4$
- $b = 2$
- $d = \log_b a = 2$
- $f(n) = n \in O(n^{d-\epsilon})$ for $\epsilon = 1$
- **Case 1** → $T(n) \in \Theta(n^2)$

Regularity condition

Examples

Two types of work:

- Solving the base cases: n^d
 - $d = \log_b a$
- Splitting/combining: $f(n)$

Comparable

- Consider a recurrence of the generic form: $T(n) = aT(n/b) + f(n)$.

Critical exponent: $d = \log_b a$

Case 1:

- $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.

▷ $T(n) \in \Theta(n^d)$

Case 2:

- $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.

▷ $T(n) \in \Theta(n^d \log^{k+1} n)$

Case 3:

- $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
- $af(n/b) \leq cf(n)$ for some $c < 1$.

▷ $T(n) \in \Theta(f(n))$

Solve: $T(n) = 2T(n/2) + n$.

- $a = 2$
- $b = 2$
- $d = \log_b a = 1$
- $f(n) = n \in \Theta(n^d \log^k n)$ for $k = 0$
- **Case 2** → $T(n) \in \Theta(n \log n)$

Regularity condition

Question 3 @ VisuAlgo online quiz

Question: $T(n) = 4T\left(\frac{n}{2}\right) + n^3$ satisfies which case of the master theorem?

$$T(n) = aT(n/b) + f(n).$$

Critical exponent: $d = \log_b a$

Case 1:

- $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.

$$\triangleright T(n) \in \Theta(n^d)$$

- Case 1.

Case 2:

- $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.

$$\triangleright T(n) \in \Theta(n^d \log^{k+1} n)$$

- Case 2.

Case 3:

- $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
- $af(n/b) \leq cf(n)$ for some $c < 1$.

$$\triangleright T(n) \in \Theta(f(n))$$

- Case 3.

- None of the above.

Regularity condition

Remarks

- The master theorem does not cover all recurrences of the generic form:

- $T(n) = aT(n/b) + f(n)$.

- **Example:**

- $T(n) = T(n/2) + 2\sqrt{\log n}$.

- Critical exponent: $d = \log_b a = 0$.
- $f(n) = 2\sqrt{\log n} \notin O(n^{0-\varepsilon}) \rightarrow$ Not Case 1.
- $f(n) = 2\sqrt{\log n} \notin \Theta(n^0 \log^k n)$ for any $k \geq 0 \rightarrow$ Not Case 2.
- $f(n) = 2\sqrt{\log n} \notin \Omega(n^{0+\varepsilon}) \rightarrow$ Not Case 3.

Remarks

- The master theorem does not cover all recurrences of the generic form:

- $T(n) = aT(n/b) + f(n)$.

- **Example:**

- $T(n) = T(n/2) + 2\sqrt{\log n}$.

- Critical exponent: $d = \log_b a = 0$.
- $f(n) = 2\sqrt{\log n} \notin O(n^{0-\varepsilon}) \rightarrow$ Not Case 1.
- $f(n) = 2\sqrt{\log n} \notin \Theta(n^0 \log^k n)$ for any $k \geq 0 \rightarrow$ Not Case 2.
- $f(n) = 2\sqrt{\log n} \notin \Omega(n^{0+\varepsilon}) \rightarrow$ Not Case 3.

- **Exercise:**

- $T(n) \in \Theta\left(2\sqrt{\log n} \cdot \sqrt{\log n}\right)$



Indeed, all three cases are not applicable.

Remarks

- The condition $f(n) \in \Omega(n^{d+\epsilon})$ is redundant in Case 3.

Critical exponent: $d = \log_b a$

Case 3:

- $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
- $af(n/b) \leq cf(n)$ for some $c < 1$.

Regularity condition

Regularity condition

$$af(n/b) \leq cf(n)$$



$$\frac{a}{c}f(n/b) \leq f(n)$$



$$\left(\frac{a}{c}\right)^i f(1) \leq f(b^i)$$



$$\Omega(n^{d+\epsilon}) \ni n^{\log_b \left(\frac{a}{c}\right)} f(1) = \left(\frac{a}{c}\right)^{\log_b n} f(1) \leq f(n)$$

$$\epsilon = \log_b \left(\frac{a}{c}\right) - \log_b a > 0 \quad n = b^i$$

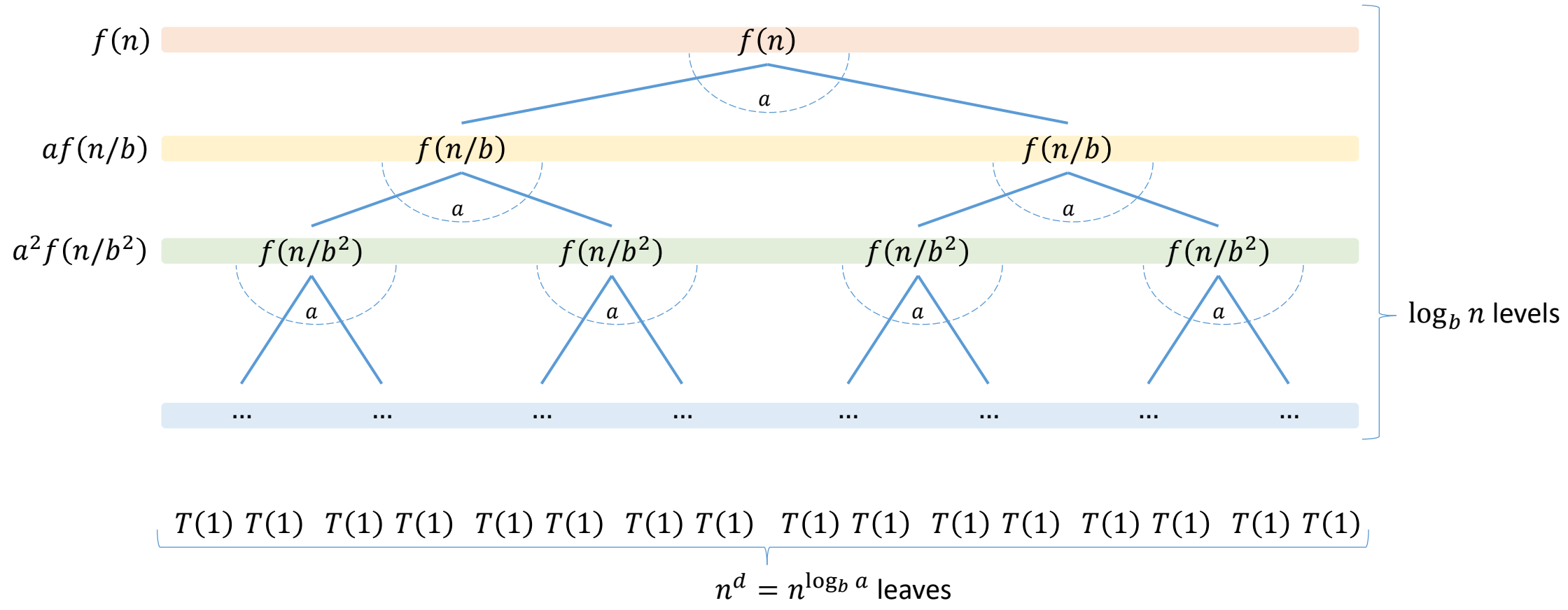
Critical exponent: $d = \log_b a$

Case 3: $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
• $af(n/b) \leq cf(n)$ for some $c < 1$.



Goal: $T(n) \in \Theta(f(n))$

Proof of the master theorem



Critical exponent: $d = \log_b a$

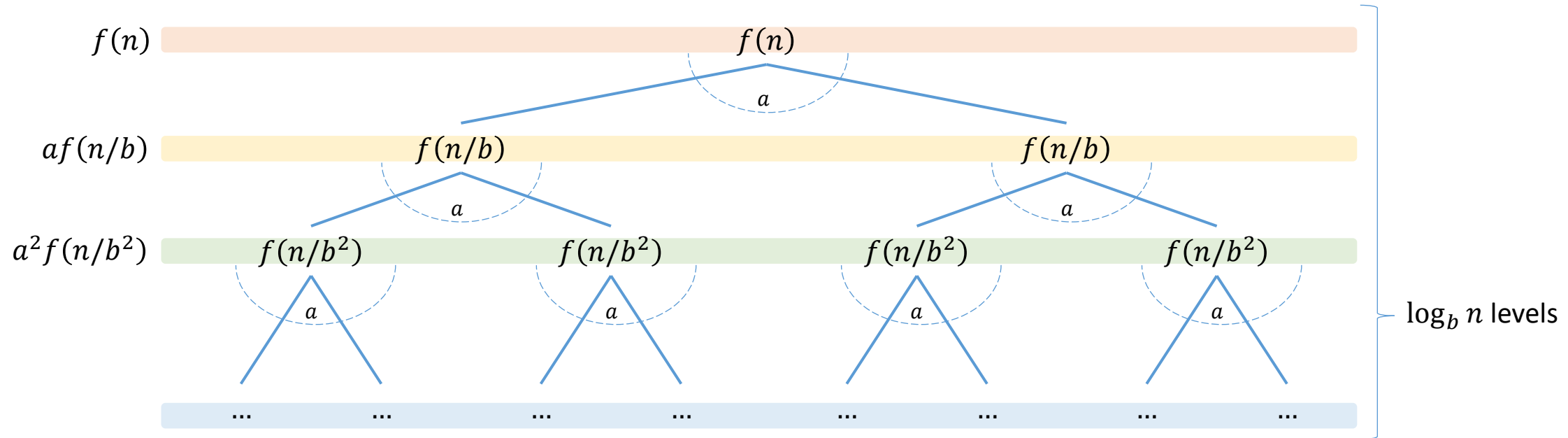
Case 3: $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.

- $af(n/b) \leq cf(n)$ for some $c < 1$.



Goal: $T(n) \in \Theta(f(n))$

Proof of the master theorem



$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$

$n^d = n^{\log_b a}$ leaves

Solving the base cases: The cost is $n^d T(1) \in o(f(n))$.

Critical exponent: $d = \log_b a$

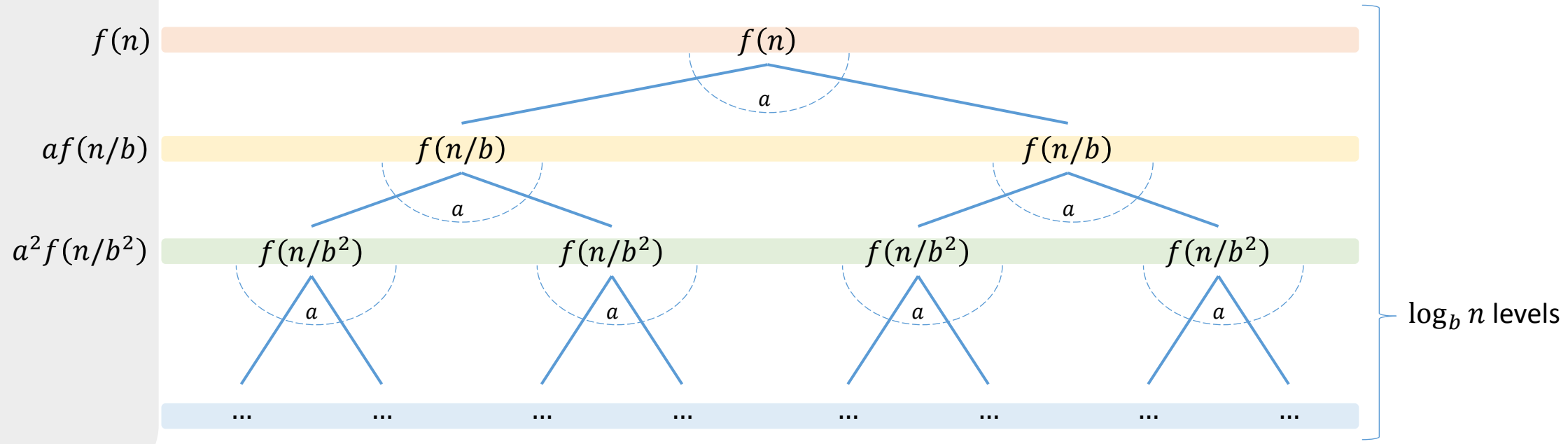
Case 3: $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
• $af(n/b) \leq cf(n)$ for some $c < 1$.



Goal: $T(n) \in \Theta(f(n))$

Proof of the master theorem

Splitting/combining:



Just need to show that this part is $\Theta(f(n))$.

$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$

$n^d = n^{\log_b a}$ leaves

Solving the base cases: The cost is $n^d T(1) \in o(f(n))$.

Critical exponent: $d = \log_b a$

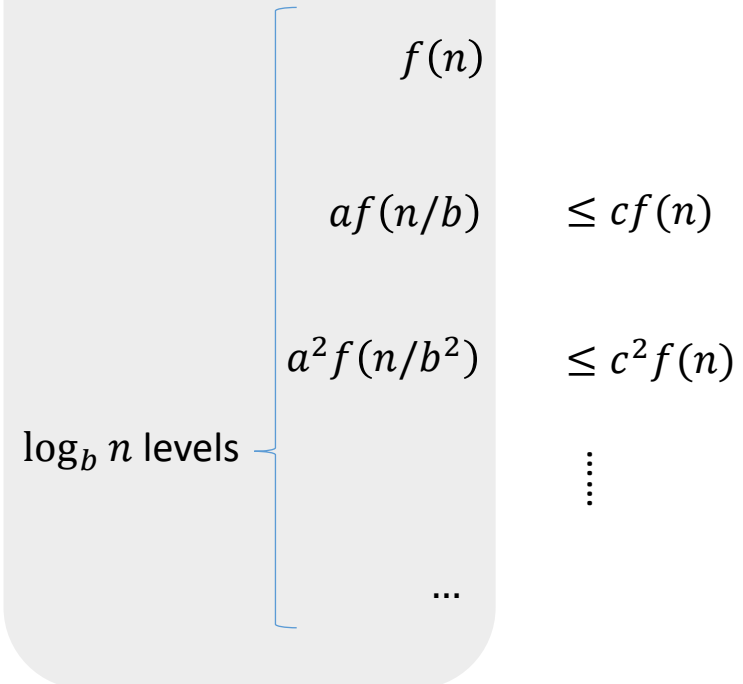
Proof of the master theorem

Case 3: $f(n) \in \Omega(n^{d+\epsilon})$ for some $\epsilon > 0$.
• $af(n/b) \leq cf(n)$ for some $c < 1$.



Goal: $T(n) \in \Theta(f(n))$

Splitting/combining:



$$(1 + c + c^2 + \dots) < \frac{1}{1 - c}$$

Just need to show that this part is $\Theta(f(n))$.

$\Omega(f(n)) \ni f(n) \leq \text{overall cost} \leq f(n) \cdot (1 + c + c^2 + \dots) \in O(f(n))$

$\underbrace{\hspace{10em}}_{\log_b n \text{ terms}}$

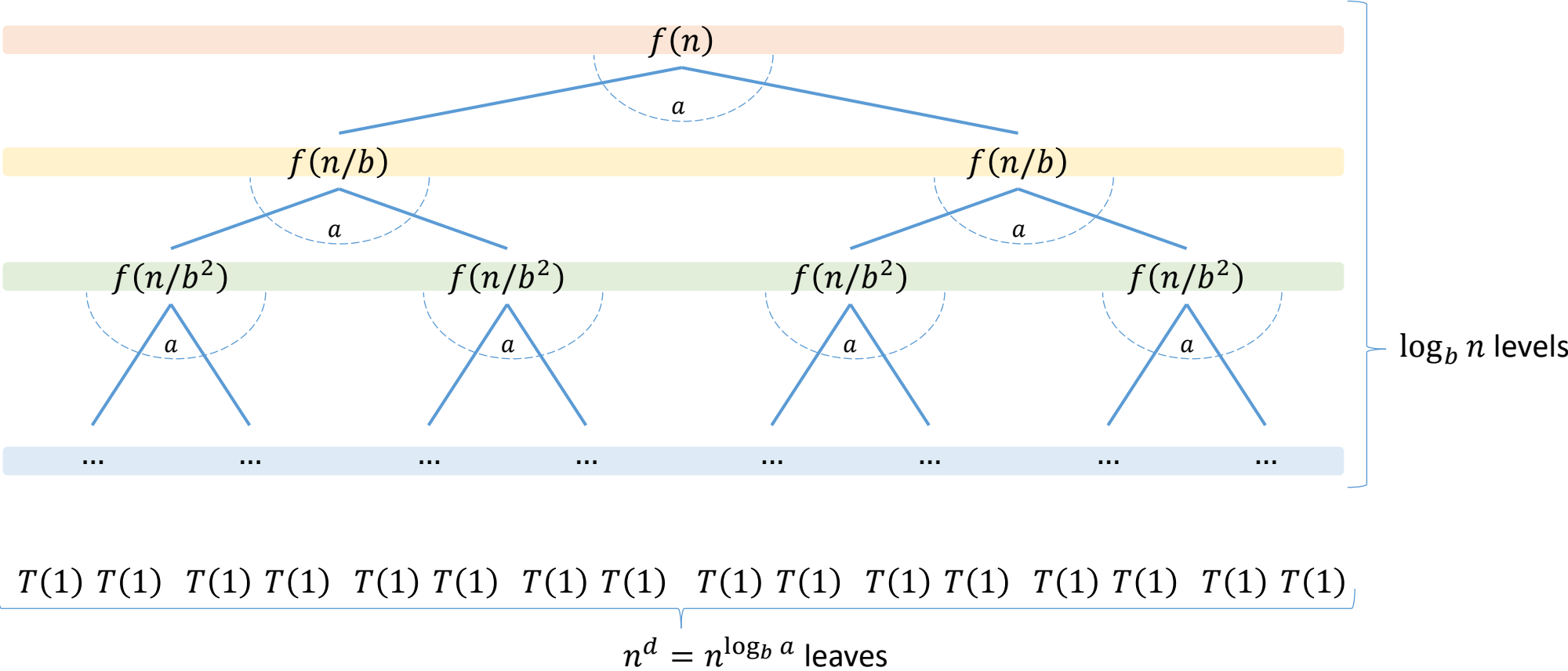
Critical exponent: $d = \log_b a$

Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.



Goal: $T(n) \in \Theta(n^d)$

Proof of the master theorem



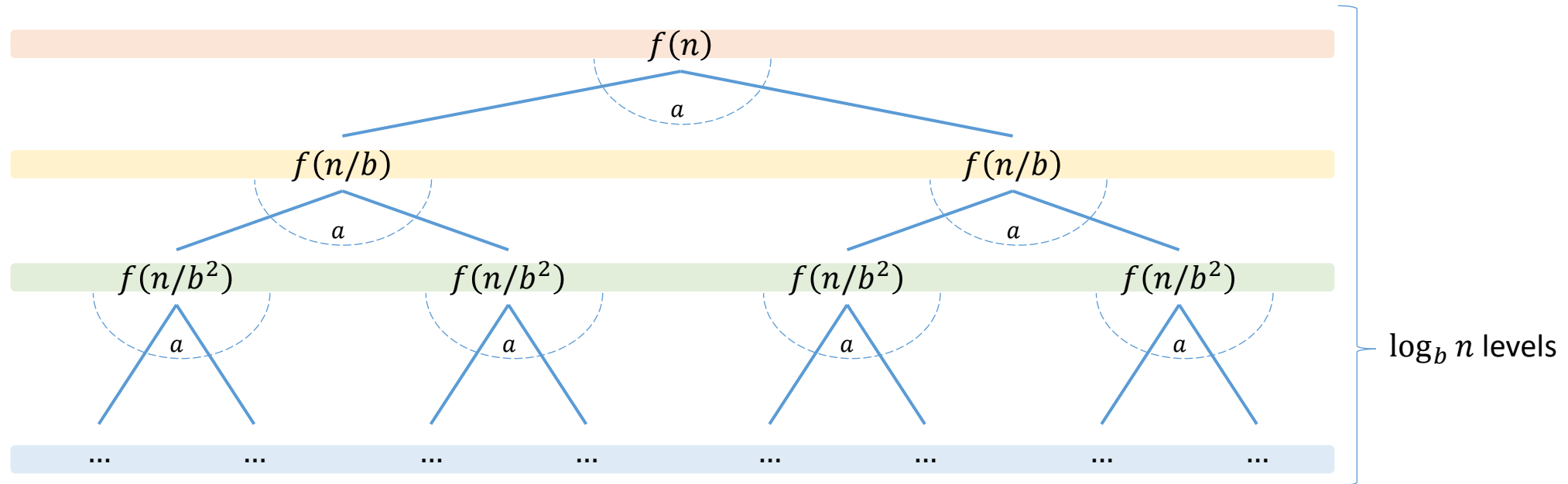
Critical exponent: $d = \log_b a$

Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.

Proof of the master theorem



Goal: $T(n) \in \Theta(n^d)$



$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$

$n^d = n^{\log_b a}$ leaves

Solving the base cases: The cost is already $n^d T(1) \in \Theta(n^d)$.

Critical exponent: $d = \log_b a$

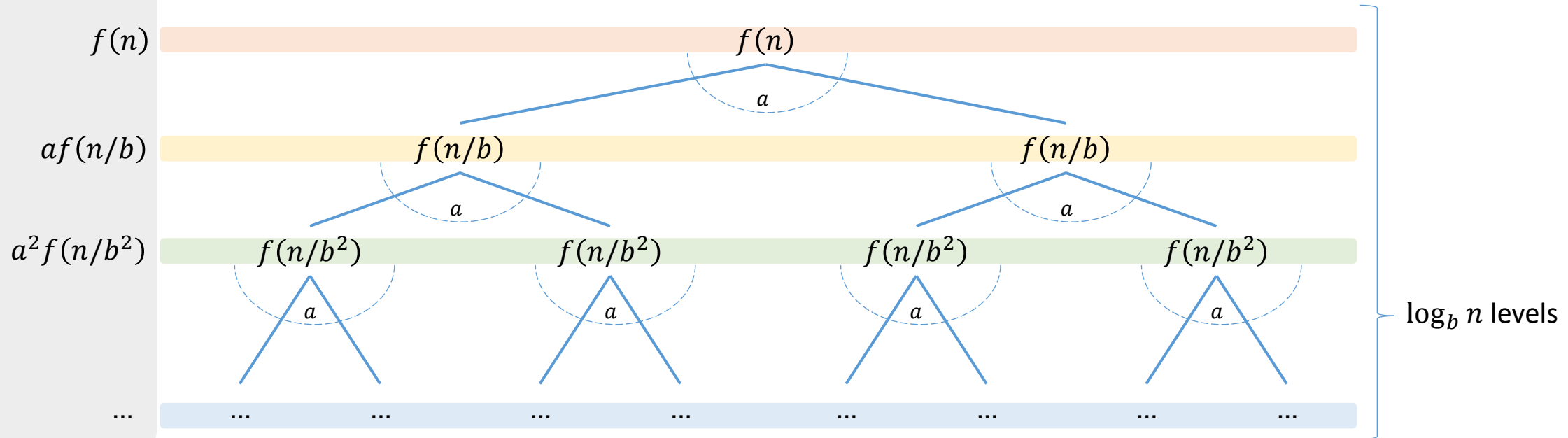
Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.

Proof of the master theorem



Goal: $T(n) \in \Theta(n^d)$

Splitting/combining:



Just need to show that this part is $O(n^d)$.

$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$

$n^d = n^{\log_b a}$ leaves

Solving the base cases: The cost is already $n^d T(1) \in \Theta(n^d)$.

Critical exponent: $d = \log_b a$

Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.



Goal: $T(n) \in \Theta(n^d)$

Proof of the master theorem

Splitting/combining:

$$f(n) \leq cn^{d-\epsilon}$$

$$af(n/b) \leq ac \left(\frac{n}{b}\right)^{d-\epsilon}$$

$$a^2 f(n/b^2) \leq a^2 c \left(\frac{n}{b^2}\right)^{d-\epsilon}$$

$\log_b n$ levels



... $\exists n_0 > 0$ and $\exists c > 0$
such that $\forall (n \geq n_0), f(n) \leq cn^{d-\epsilon}$



Just need to show that
this part is $O(n^d)$.

Critical exponent: $d = \log_b a$

Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.



Goal: $T(n) \in \Theta(n^d)$

Proof of the master theorem

Splitting/combining:

$\log_b n$ levels

$$\begin{aligned} f(n) &\leq cn^{d-\epsilon} \\ af(n/b) &\leq ac \left(\frac{n}{b}\right)^{d-\epsilon} = cn^{d-\epsilon} \cdot ab^{\epsilon-d} \\ a^2 f(n/b^2) &\leq a^2 c \left(\frac{n}{b^2}\right)^{d-\epsilon} = cn^{d-\epsilon} \cdot (ab^{\epsilon-d})^2 \\ &\vdots \end{aligned}$$

\triangle

$\exists n_0 > 0$ and $\exists c > 0$
such that $\forall (n \geq n_0), f(n) \leq cn^{d-\epsilon}$



Just need to show that
this part is $O(n^d)$.

Critical exponent: $d = \log_b a$

Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.



Goal: $T(n) \in \Theta(n^d)$

Proof of the master theorem

Splitting/combining:

$$f(n) \leq cn^{d-\epsilon}$$

$$af(n/b) \leq ac \left(\frac{n}{b}\right)^{d-\epsilon} = cn^{d-\epsilon} \cdot ab^{\epsilon-d} = cn^{d-\epsilon} \cdot b^\epsilon$$

$$a^2 f(n/b^2) \leq a^2 c \left(\frac{n}{b^2}\right)^{d-\epsilon} = cn^{d-\epsilon} \cdot (ab^{\epsilon-d})^2 = cn^{d-\epsilon} \cdot b^{2\epsilon}$$

$\log_b n$ levels



$\exists n_0 > 0$ and $\exists c > 0$
such that $\forall (n \geq n_0), f(n) \leq cn^{d-\epsilon}$



$$b^d = a$$



Just need to show that
this part is $O(n^d)$.

Critical exponent: $d = \log_b a$

Case 1: $f(n) \in O(n^{d-\epsilon})$ for some $\epsilon > 0$.



Goal: $T(n) \in \Theta(n^d)$

Proof of the master theorem

Splitting/combining:

$$f(n) \leq cn^{d-\epsilon}$$

$$af(n/b) \leq ac \left(\frac{n}{b}\right)^{d-\epsilon} = cn^{d-\epsilon} \cdot ab^{\epsilon-d} = cn^{d-\epsilon} \cdot b^\epsilon$$

$$a^2 f(n/b^2) \leq a^2 c \left(\frac{n}{b^2}\right)^{d-\epsilon} = cn^{d-\epsilon} \cdot (ab^{\epsilon-d})^2 = cn^{d-\epsilon} \cdot b^{2\epsilon}$$

$\log_b n$ levels



$\exists n_0 > 0$ and $\exists c > 0$
such that $\forall (n \geq n_0), f(n) \leq cn^{d-\epsilon}$



$$b^d = a$$



Just need to show that this part is $O(n^d)$.

Overall cost:

$$O(n^{d-\epsilon}) \cdot (1 + b^\epsilon + b^{2\epsilon} + \dots) = O(n^{d-\epsilon}) \cdot O(b^{\epsilon \log_b n}) = O(n^{d-\epsilon}) \cdot O(n^\epsilon) = O(n^d)$$

$\log_b n$ terms

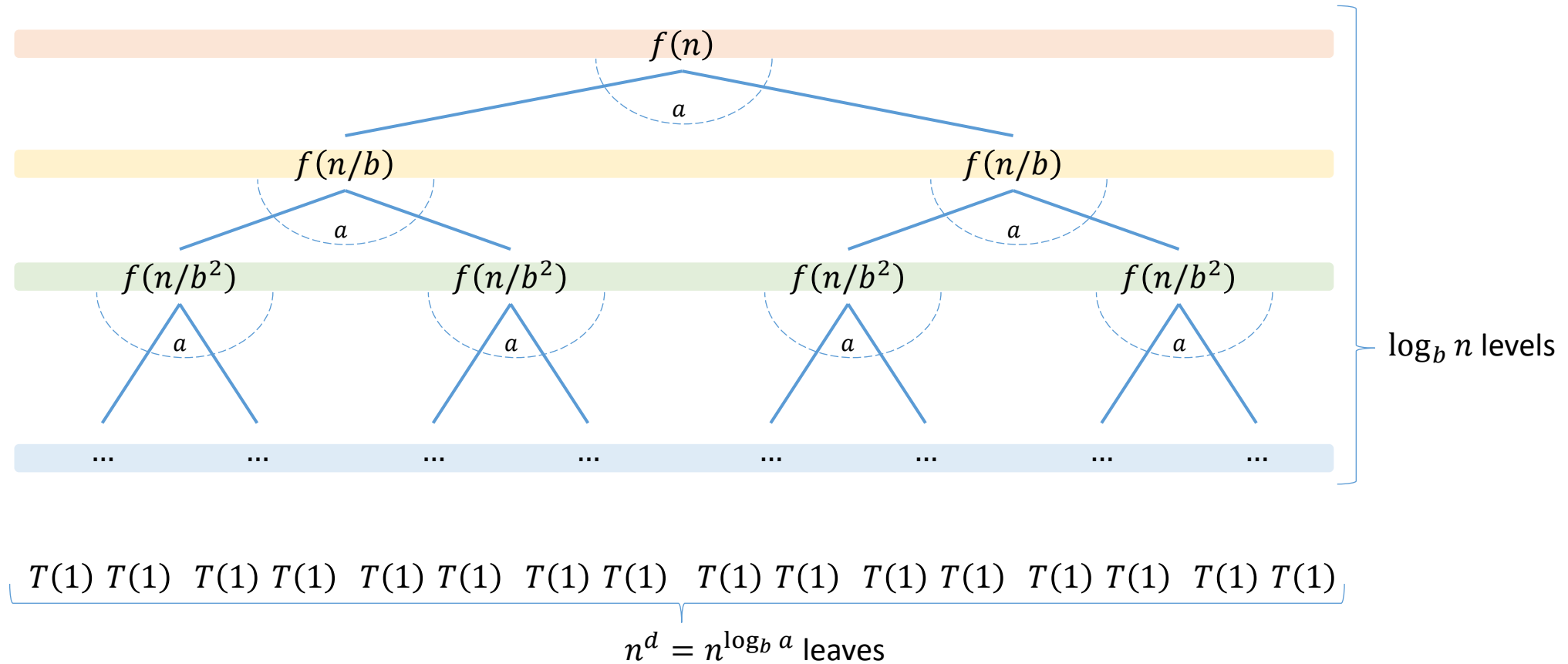
Critical exponent: $d = \log_b a$

Case 2: $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.

Proof of the master theorem



Goal: $T(n) \in \Theta(n^d \log^{k+1} n)$



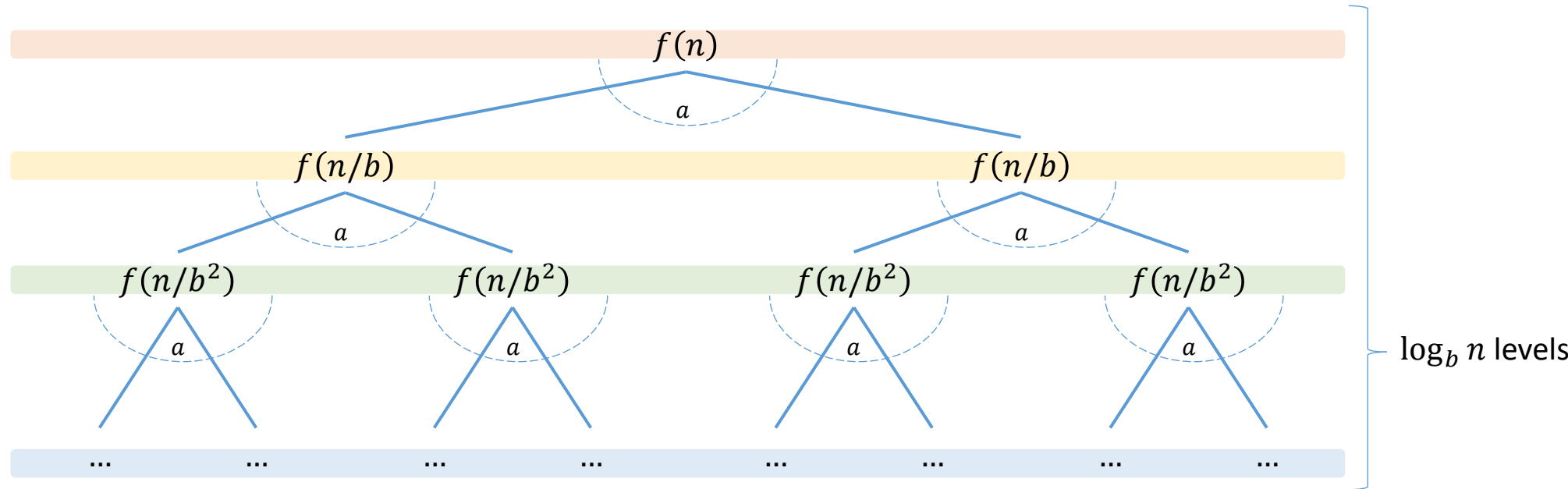
Critical exponent: $d = \log_b a$

Case 2: $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.

Proof of the master theorem



Goal: $T(n) \in \Theta(n^d \log^{k+1} n)$



$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$
 $n^d = n^{\log_b a}$ leaves

Solving the base cases: The cost is $n^d T(1) \in o(n^d \log^{k+1} n)$.

Critical exponent: $d = \log_b a$

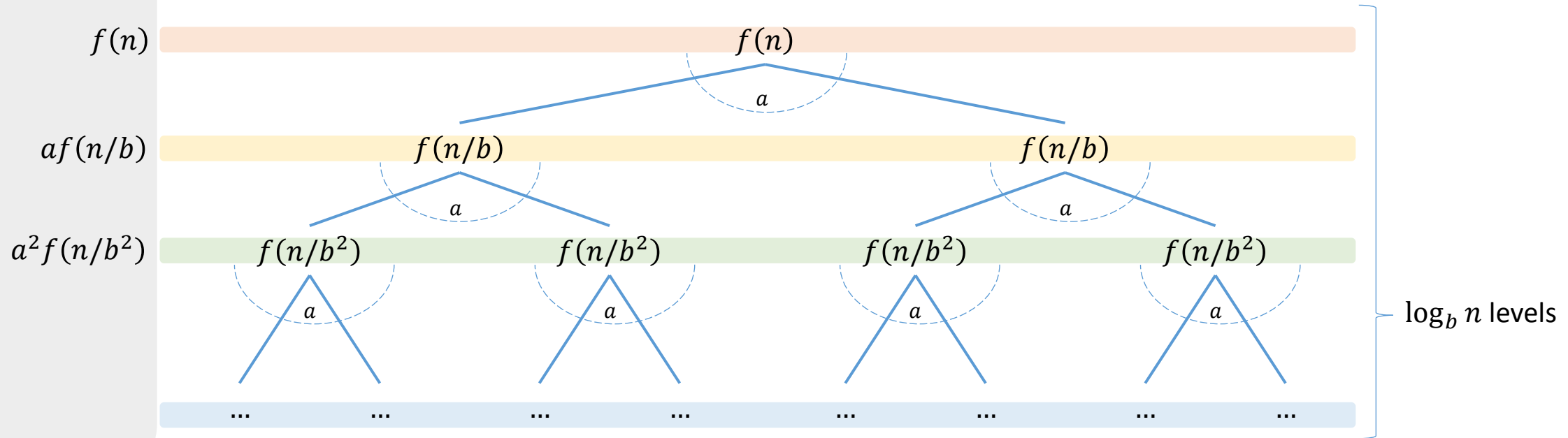
Case 2: $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.



Goal: $T(n) \in \Theta(n^d \log^{k+1} n)$

Proof of the master theorem

Splitting/combining:



Need to show that this part is $\Theta(n^d \log^{k+1} n)$.

$T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)$

$n^d = n^{\log_b a}$ leaves

Solving the base cases: The cost is $n^d T(1) \in o(n^d \log^{k+1} n)$.

Critical exponent: $d = \log_b a$

Case 2: $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.



Goal: $T(n) \in \Theta(n^d \log^{k+1} n)$

Proof of the master theorem

Splitting/combining:

$$f(n)$$

$$\in \Theta(n^{\log_b a} \log^k n) = \Theta(n^d \log^k n)$$

$$af(n/b)$$

$$\in a \cdot \Theta\left(\left(\frac{n}{b}\right)^{\log_b a} \log^k \frac{n}{b}\right) = \Theta\left(n^{\log_b a} \log^k \frac{n}{b}\right) = \Theta\left(n^d \log^k \frac{n}{b}\right)$$

$$a^2 f(n/b^2)$$

$$\in a^2 \cdot \Theta\left(\left(\frac{n}{b^2}\right)^{\log_b a} \log^k \frac{n}{b^2}\right) = \Theta\left(n^{\log_b a} \log^k \frac{n}{b^2}\right) = \Theta\left(n^d \log^k \frac{n}{b^2}\right)$$

$\log_b n$ levels

⋮

...



Need to show that this part is $\Theta(n^d \log^{k+1} n)$.

Critical exponent: $d = \log_b a$

Case 2: $f(n) \in \Theta(n^d \log^k n)$ for some $k \geq 0$.



Goal: $T(n) \in \Theta(n^d \log^{k+1} n)$

Proof of the master theorem

Splitting/combining:

$$f(n)$$

$$\in \Theta(n^{\log_b a} \log^k n) = \Theta(n^d \log^k n)$$

$$af(n/b)$$

$$\in a \cdot \Theta\left(\left(\frac{n}{b}\right)^{\log_b a} \log^k \frac{n}{b}\right) = \Theta\left(n^{\log_b a} \log^k \frac{n}{b}\right) = \Theta\left(n^d \log^k \frac{n}{b}\right)$$

$$a^2 f(n/b^2)$$

$$\in a^2 \cdot \Theta\left(\left(\frac{n}{b^2}\right)^{\log_b a} \log^k \frac{n}{b^2}\right) = \Theta\left(n^{\log_b a} \log^k \frac{n}{b^2}\right) = \Theta\left(n^d \log^k \frac{n}{b^2}\right)$$

⋮

$\log_b n$ levels

...

$$\log^k n + \log^k \frac{n}{b} + \log^k \frac{n}{b^2} + \dots = (\log n)^k + (\log n - \log b)^k + (\log n - 2 \log b)^k + \dots \in \Theta((\log n)^{k+1})$$

$$\text{Overall cost: } \Theta(n^d) \cdot \left(\log^k n + \log^k \frac{n}{b} + \log^k \frac{n}{b^2} + \dots\right) = \Theta(n^d \log^{k+1} n)$$

$\log_b n$ terms, half of them being at least $\left(\frac{1}{2}\right)^k \log^k n$

Need to show that this part is $\Theta(n^d \log^{k+1} n)$.



Floors and ceilings

- In the master theorem, floors and ceilings within the recursive subproblem sizes do not affect the asymptotic growth of the function.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases} \quad T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) & \text{if } n > 1 \end{cases}$$

- **Optional readings:**

- Section 4.7 of CLRS 4e “Akra–Bazzi recurrences.”
- William Kuszmaul and Charles E. Leiserson. “Floors and Ceilings in Divide-and-Conquer Recurrences.” Symposium on Simplicity in Algorithms (SOSA 2021).
<https://epubs.siam.org/doi/10.1137/1.9781611976496.15>

Acknowledgement

- The slides are modified from previous editions of this course and similar course elsewhere.
- **List of credits:**
 - Surender Baswana
 - Diptarka Chakraborty
 - Yi-Jun Chang
 - Erik Demaine
 - Steven Halim
 - Sanjay Jain
 - Wee Sun Lee
 - Charles Leiserson
 - Hon Wai Leong
 - Warut Sukhompong
 - Wing-Kin Sung