CS3230 – Design and Analysis of Algorithms (S1 AY2024/25)

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.

Time complexity

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

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.

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

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

Merge sort

$T(n)$ → **MergeSort**$(A[1..n])$

- If $n \geq 2$, do the following steps.
  - **MergeSort**$(A[1..\lceil n/2 \rceil])$.
  - **MergeSort**$(A[\lceil n/2 \rceil + 1..n])$.
  - "Merge" the two sorted arrays.

We omit the details.

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

VisuAlgo (Merge sort): https://visualgo.net/en/sorting?mode=Merge

Merge sort

$T(n)$ → **MergeSort**$(A[1..n])$

- If $n \geq 2$, do the following steps. ← $\Theta(1)$
  - **MergeSort**$(A[1..\lceil n/2 \rceil])$. ← $T(n/2)$
  - **MergeSort**$(A[\lceil n/2 \rceil + 1..n])$. ← $T(n/2)$
  - "Merge" the two sorted arrays.

Here we approximate $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$ by $n/2$.

- This is slightly sloppy.
- As we will later discuss, removing floors and ceilings does not affect the asymptotic complexity in most of the cases.

VisuAlgo (Merge sort): https://visualgo.net/en/sorting?mode=Merge

Merge sort

$T(n)$ → **MergeSort**$(A[1..n])$

- If $n \geq 2$, do the following steps. ← $\Theta(1)$
  - **MergeSort**$(A[1..\lceil n/2 \rceil])$. ← $T(n/2)$
  - **MergeSort**$(A[\lceil n/2 \rceil + 1..n])$. ← $T(n/2)$
  - "Merge" the two sorted arrays. ← $\Theta(n)$

[**1**, 5, 7],	[**2**, 4, 6],	[ ]
[1, **5**, 7],	[**2**, 4, 6],	[1]
[1, **5**, 7],	[2, **4**, 6],	[1, 2]
[1, **5**, 7],	[2, 4, **6**],	[1, 2, 4]
[1, 5, **7**],	[2, 4, **6**],	[1, 2, 4, 5]
[1, 5, **7**],	[2, 4, 6],	[1, 2, 4, 5, 6]
[1, 5, 7],	[2, 4, 6],	[1, 2, 4, 5, 6, 7]

VisuAlgo (Merge sort): https://visualgo.net/en/sorting?mode=Merge

Merge sort

$T(n)$ → **MergeSort**$(A[1..n])$

- If $n \geq 2$, do the following steps. ← $\Theta(1)$
  - **MergeSort**$(A[1..\lceil n/2 \rceil])$. ← $T(n/2)$
  - **MergeSort**$(A[\lceil n/2 \rceil + 1..n])$. ← $T(n/2)$
  - "Merge" the two sorted arrays. ← $\Theta(n)$

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

VisuAlgo (Merge sort): https://visualgo.net/en/sorting?mode=Merge

Merge sort

$T(n)$ → **MergeSort**$(A[1..n])$

- If $n \geq 2$, do the following steps. ← $\Theta(1)$
  - **MergeSort**$(A[1..\lceil n/2 \rceil])$. ← $T(n/2)$
  - **MergeSort**$(A[\lceil n/2 \rceil + 1..n])$. ← $T(n/2)$
  - "Merge" the two sorted arrays. ← $\Theta(n)$

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

Base case

Inductive step

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.

VisuAlgo (Merge sort): https://visualgo.net/en/sorting?mode=Merge

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

Remarks:

If $f(n) \in O(n)$, then there exist two constant $c > 0$ 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 constant $c > 0$ and $n_0 > 0$ such that $f(n) \leq cn$ 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:**

$$\sum_{k=1}^{n} \frac{1}{k(k+1)} = \sum_{k=1}^{n} \left(\frac{1}{k} - \frac{1}{k+1} \right)$$

$$= \left(\frac{1}{1} - \frac{1}{2} \right) + \left(\frac{1}{2} - \frac{1}{3} \right) + \cdots + \left(\frac{1}{n-1} - \frac{1}{n} \right) + \left(\frac{1}{n} - \frac{1}{n+1} \right)$$

$$= 1 - \frac{1}{n+1}$$

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

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

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

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

$\vdots$

$\log n$

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

Telescoping method

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

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

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

$\log n$

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

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

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

$\vdots$

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

Telescoping method

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

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

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

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

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

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

$\log n$

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

Substitution method

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

Substitution method

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

Substitution method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} c & \text{if } n \leq 1 \\ 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & \text{if } n > 1 \end{cases}$
- **Induction hypothesis:** $T(n) \leq (c+1)n^2 - n$.

△

Guessing an upper bound of $T(n)$.
If we can prove this for all $n \in \{1, 2, 3, \dots\}$, then $T(n) \in O(n^2)$.

Substitution method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} c & \text{if } n \leq 1 \\ 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & \text{if } n > 1 \end{cases}$
- **Induction hypothesis:** $T(n) \leq (c+1)n^2 - n$.

Base case: $n = 1$.

- If $n = 1$, then $T(n) = c = (c+1)n^2 - n$.

Substitution method

- **Goal:** Solve the recurrence $T(n) = \begin{cases} c & \text{if } n \leq 1 \\ 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & \text{if } n > 1 \end{cases}$
- **Induction hypothesis:** $T(n) \leq (c+1)n^2 - n$.

Base case: $n = 1$.

- If $n = 1$, then $T(n) = c = (c+1)n^2 - n$.

Inductive step: $n \geq 2$.

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

Induction hypothesis: $T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \leq (c+1)\left\lfloor \frac{n}{2} \right\rfloor^2 - \left\lfloor \frac{n}{2} \right\rfloor$.

$$\leq 4(c+1)\left\lfloor \frac{n}{2} \right\rfloor^2 - 4\left\lfloor \frac{n}{2} \right\rfloor + n$$

The function $(c+1)x^2 - x$ is increasing when $x \geq 0$.

$$\leq 4(c+1)\left(\frac{n}{2}\right)^2 - 4\left(\frac{n}{2}\right) + n$$

$$= (c+1)n^2 - n$$

Therefore, $T(n) \in O(n^2)$.

A common mistake

- **Goal:** Solve the recurrence $T(n) = \begin{cases} c & \text{if } n \leq 1 \\ 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & \text{if } n > 1 \end{cases}$
- **Induction hypothesis:** $T(n) \leq cn^2$.

Base case: $n = 1$.

- If $n = 1$, then $T(n) = c = cn^2$.

Induction hypothesis: $T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \leq c\left\lfloor \frac{n}{2} \right\rfloor^2$.

Inductive step: $n \geq 2$.

$$T(n) = 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$
$$= 4c\left\lfloor \frac{n}{2} \right\rfloor^2 + n$$
$$\in O(n^2)$$

Incorrect proof!

You need to show that $T(n) \leq cn^2$.

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., $1, 2, 4, 8, 16, \ldots$

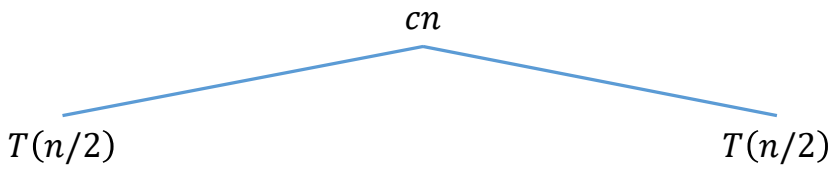
```
if (n == 1) /* base case */
  return 1;
else /* recursive caseS */
  return f(n/2) + f(n/2);
```

VisuAlgo (Recursion tree): https://visualgo.net/en/recursion

Recursion tree

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

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



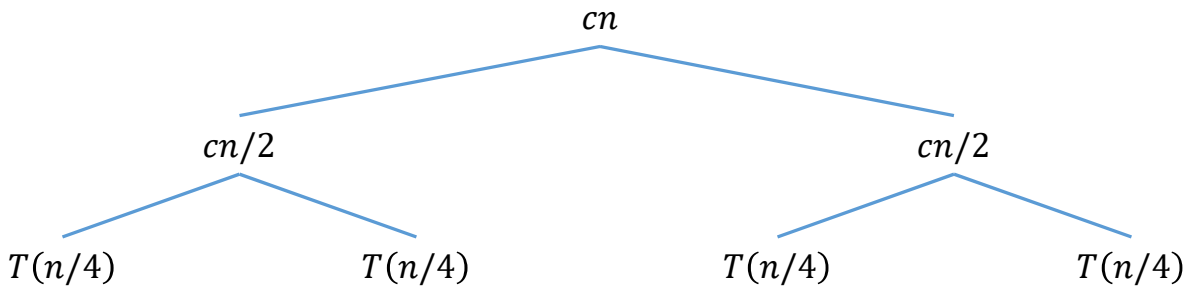
VisuAlgo (Recursion tree): https://visualgo.net/en/recursion

Recursion tree

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

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

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



VisuAlgo (Recursion tree): https://visualgo.net/en/recursion

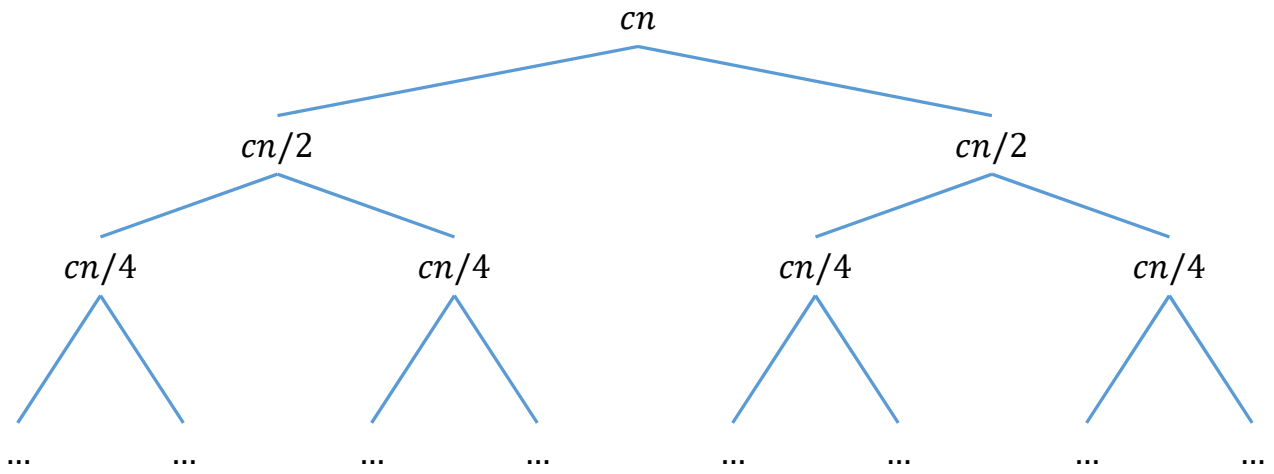
Recursion tree

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

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

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

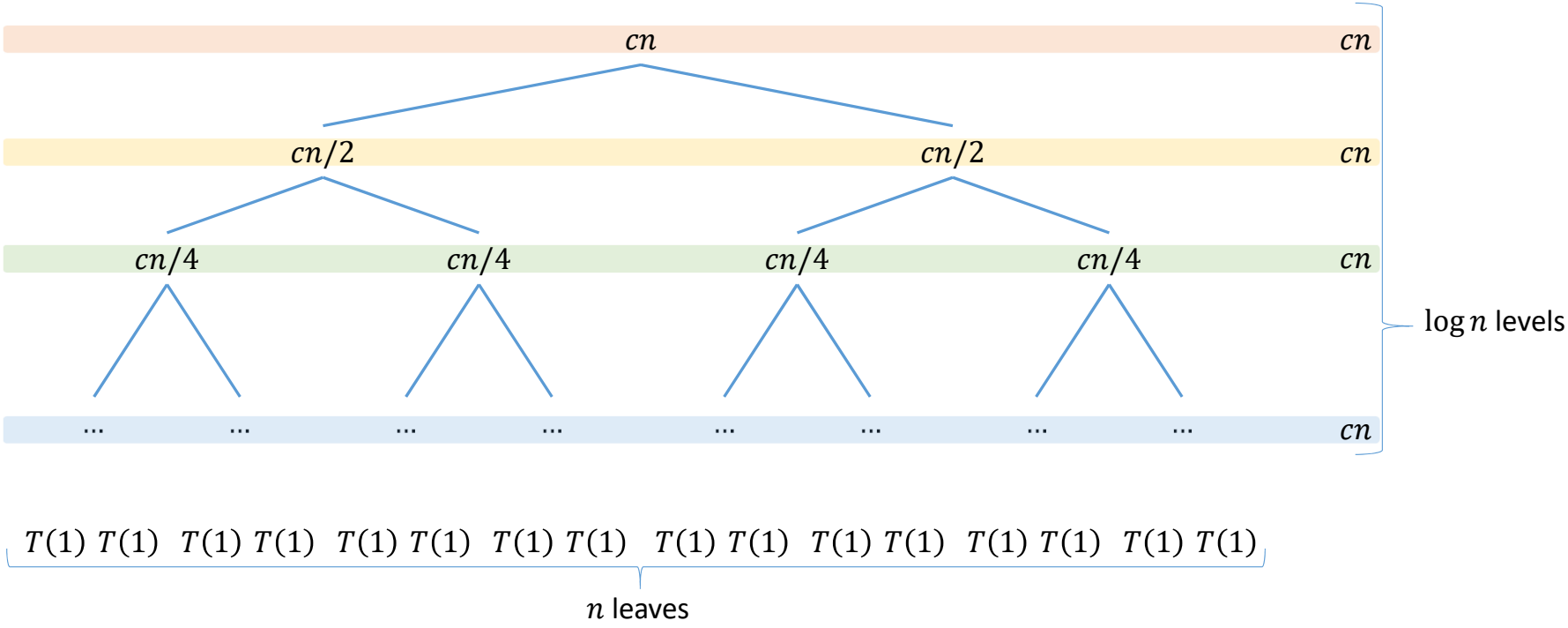
$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{cn}{4}$$



VisuAlgo (Recursion tree): https://visualgo.net/en/recursion

Recursion tree

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

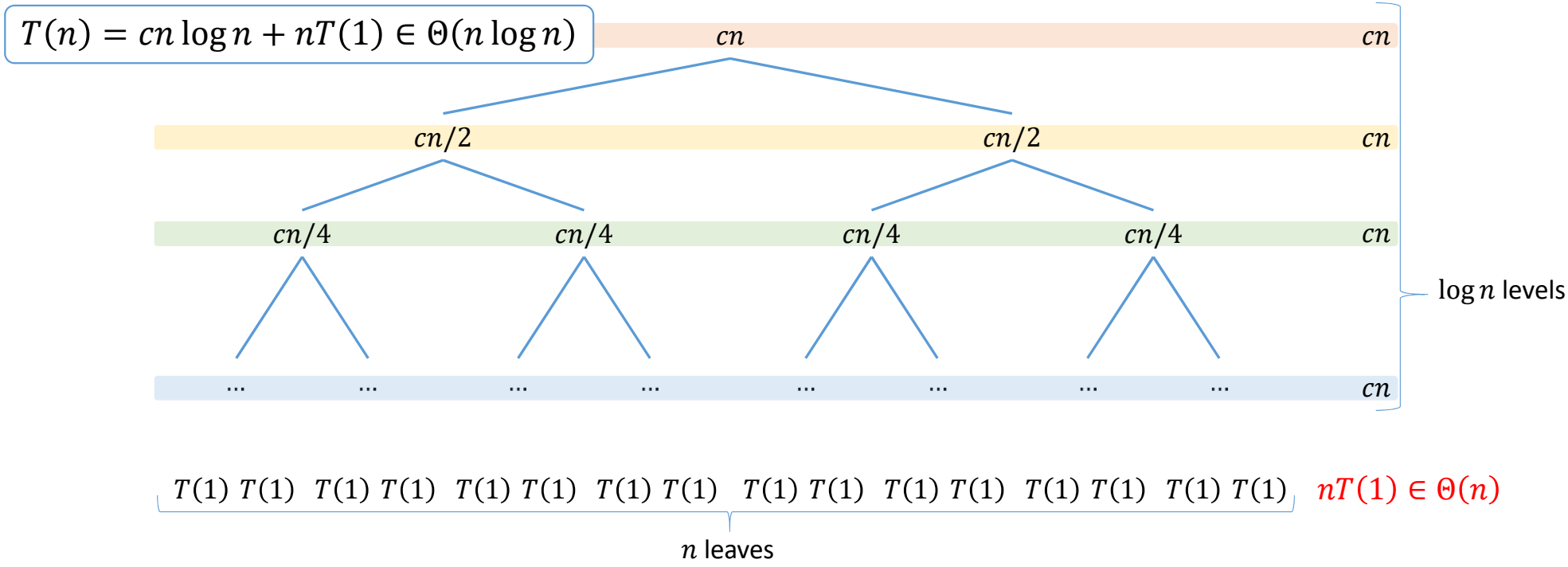


VisuAlgo (Recursion tree): https://visualgo.net/en/recursion

Recursion tree

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

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



VisuAlgo (Recursion tree): https://visualgo.net/en/recursion

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.

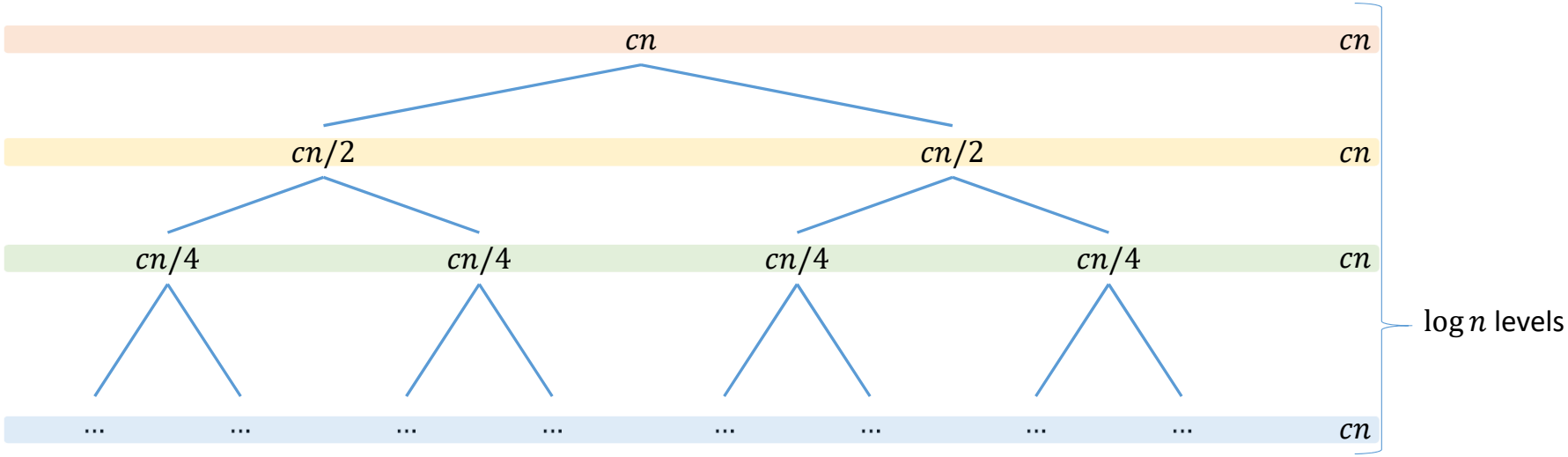
Two types of work

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

Splitting/combining:

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

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



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

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

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

▷ $T(n) \in \Theta\left(n^d\right)$

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 $c < 1$.

Regularity condition

▷ $T(n) \in \Theta(f(n))$

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** → $T(n) \in \Theta(n^2)$

VisuAlgo (Master theorem): https://visualgo.net/en/recursion?example=MT1L2 – you can change $n, a, b, f(n)$

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 $\epsilon > 0$.

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

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

Regularity condition

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

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

VisuAlgo (Master theorem): https://visualgo.net/en/recursion?example=MT2L2 – you can change $n, a, b, f(n)$

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 $\epsilon > 0$.

▷ $T(n) \in \Theta(n^d)$

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 $c < 1$.

▷ $T(n) \in \Theta(f(n))$

Regularity condition

- Case 1.
- Case 2.
- Case 3.
- None of the above.

VisuAlgo (Master theorem): https://visualgo.net/en/recursion?example=MT3L2 – you can change $n, a, b, f(n)$

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: $d = \log_b a = 0$.
- $f(n) = 2^{\sqrt{\log n}} \notin O(n^{0-\varepsilon})$ → Not Case 1.
- $f(n) = 2^{\sqrt{\log n}} \notin \Theta\left(n^0 \log^k n\right)$ for any $k \geq 0$ → Not Case 2.
- $f(n) = 2^{\sqrt{\log n}} \notin \Omega(n^{0+\varepsilon})$ → 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: $d = \log_b a = 0$.
- $f(n) = 2^{\sqrt{\log n}} \notin O(n^{0-\varepsilon})$ → Not Case 1.
- $f(n) = 2^{\sqrt{\log n}} \notin \Theta\left(n^0 \log^k n\right)$ for any $k \geq 0$ → Not Case 2.
- $f(n) = 2^{\sqrt{\log n}} \notin \Omega(n^{0+\varepsilon})$ → 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 $f(n) \in \Omega\left(n^{d+\epsilon}\right)$ is redundant in Case 3.

Critical exponent: $d = \log_b a$

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

Regularity condition

Regularity condition

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

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

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

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

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

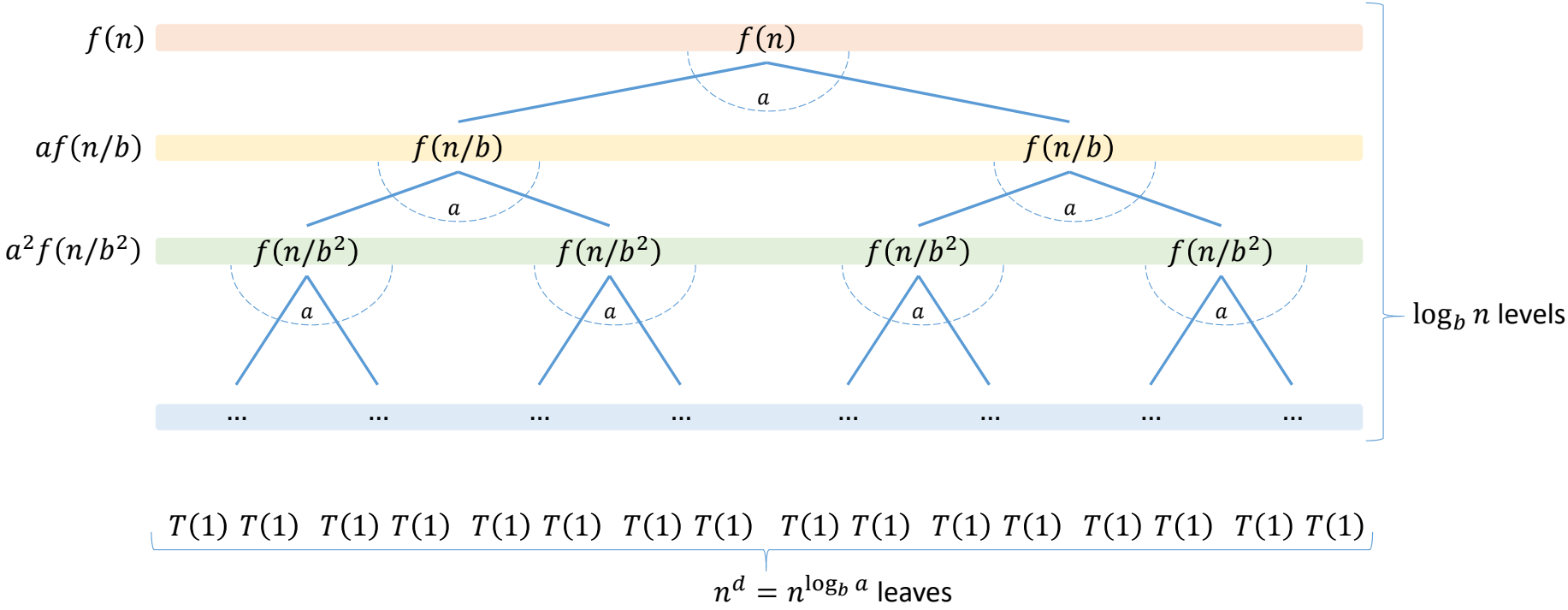
Critical exponent: $d = \log_b a$

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

▽

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

Proof of the master theorem

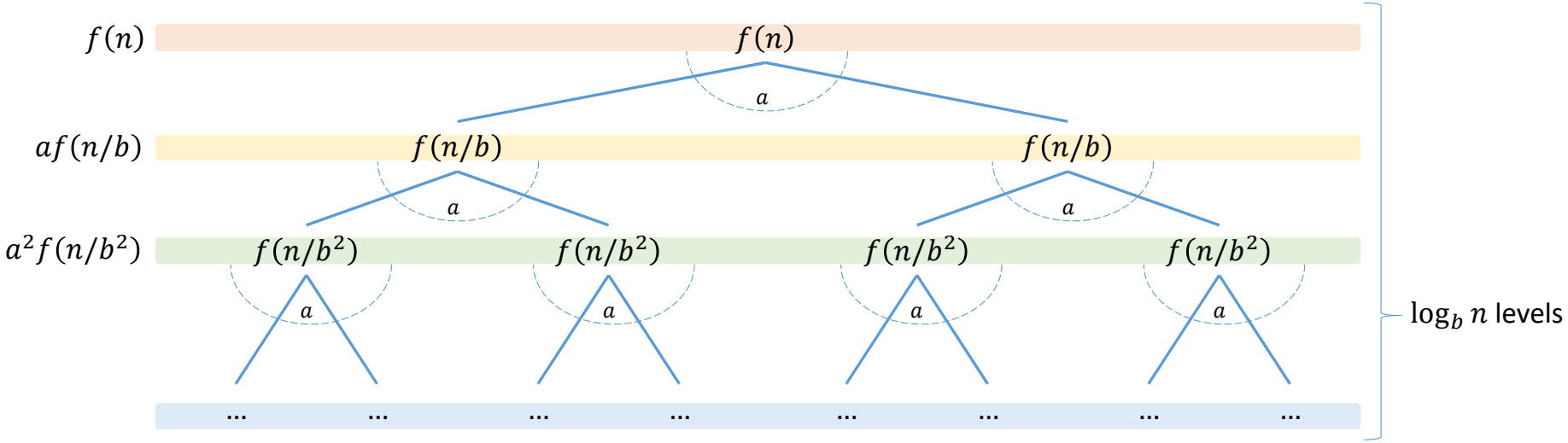


Critical exponent: $d = \log_b a$

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



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

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

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

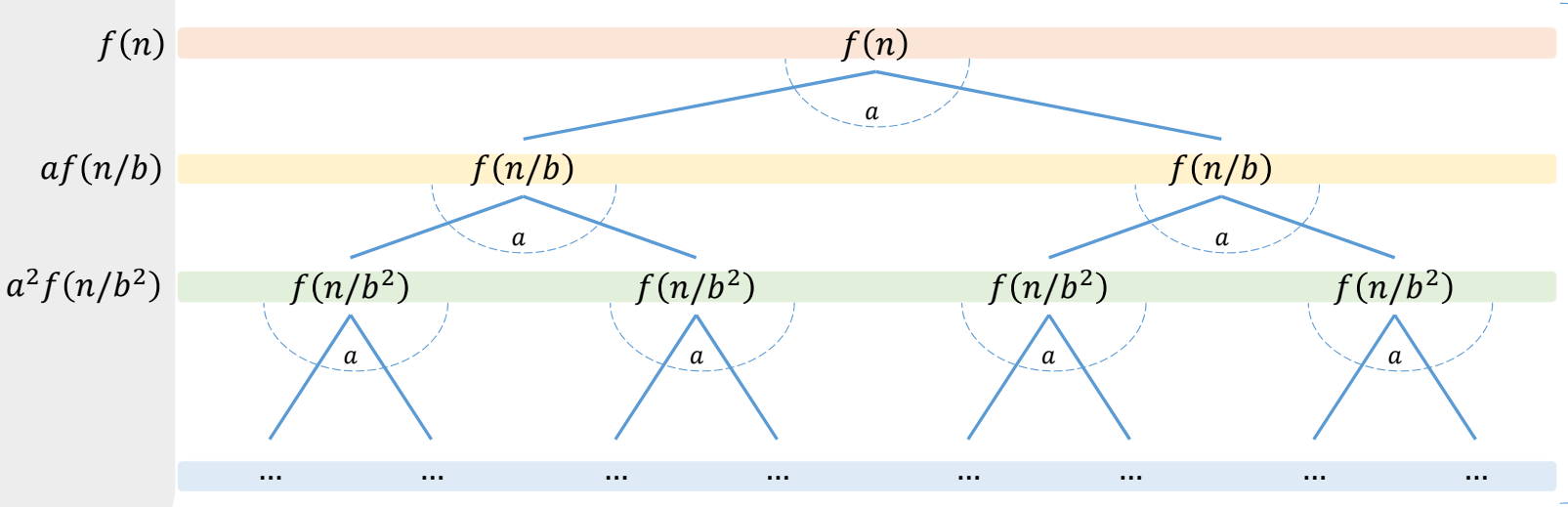
Critical exponent: $d = \log_b a$

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 $\Theta(f(n))$.

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

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

$f(n)$

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

$a^2 f(n/b^2)$ $\leq c^2 f(n)$

$\log_b n$ levels

...

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

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

$\Omega(f(n)) \ni f(n) \leq$ **overall cost** $\leq f(n) \cdot (1 + c + c^2 + \cdots) \in O(f(n))$

$\log_b n$ terms

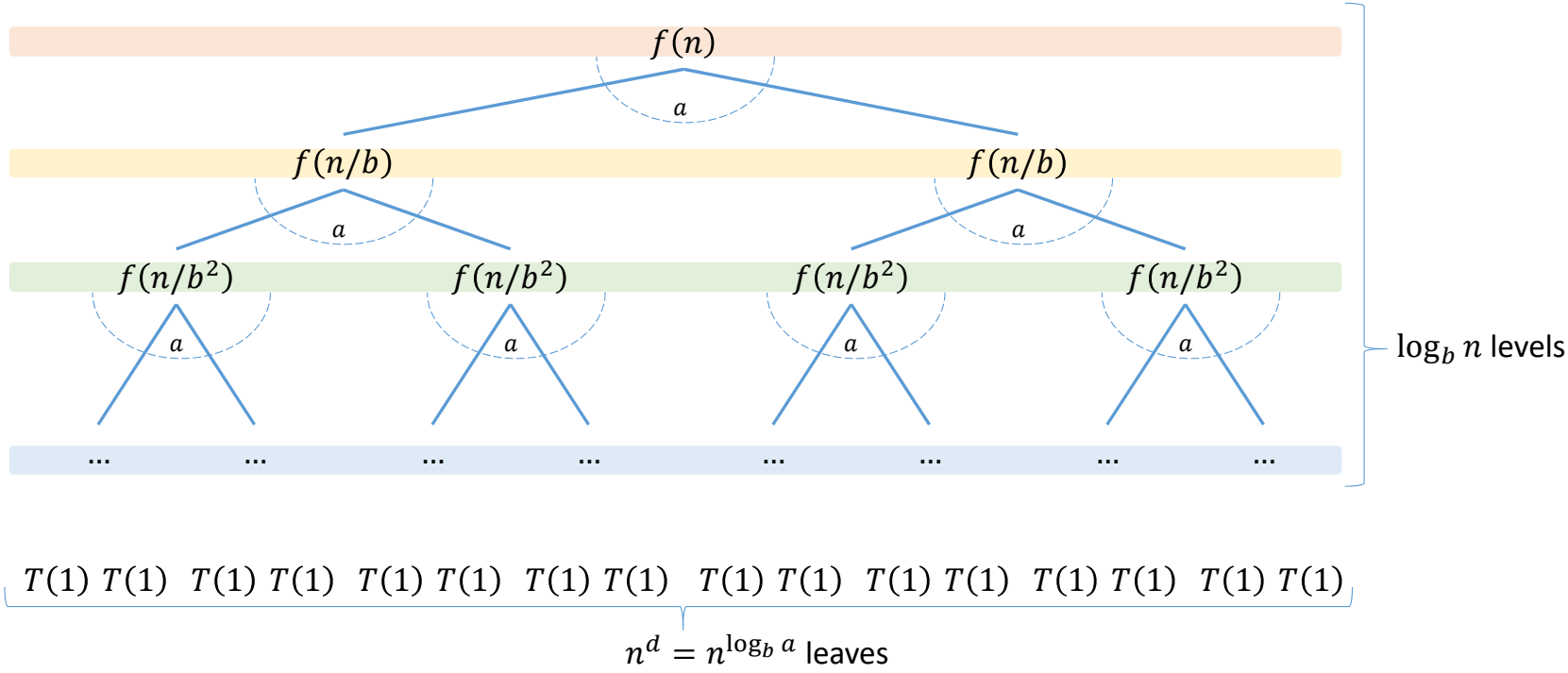
Proof of the master theorem

Critical exponent: $d = \log_b a$

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

▽

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



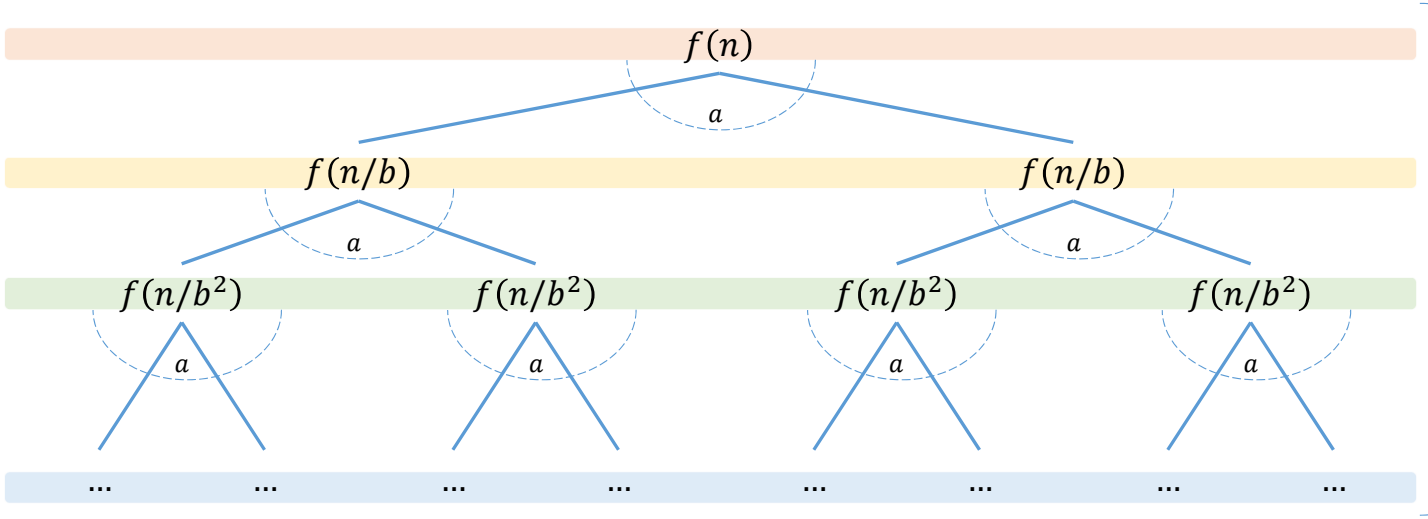
Proof of the master theorem

Critical exponent: $d = \log_b a$

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

▽

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



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

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

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

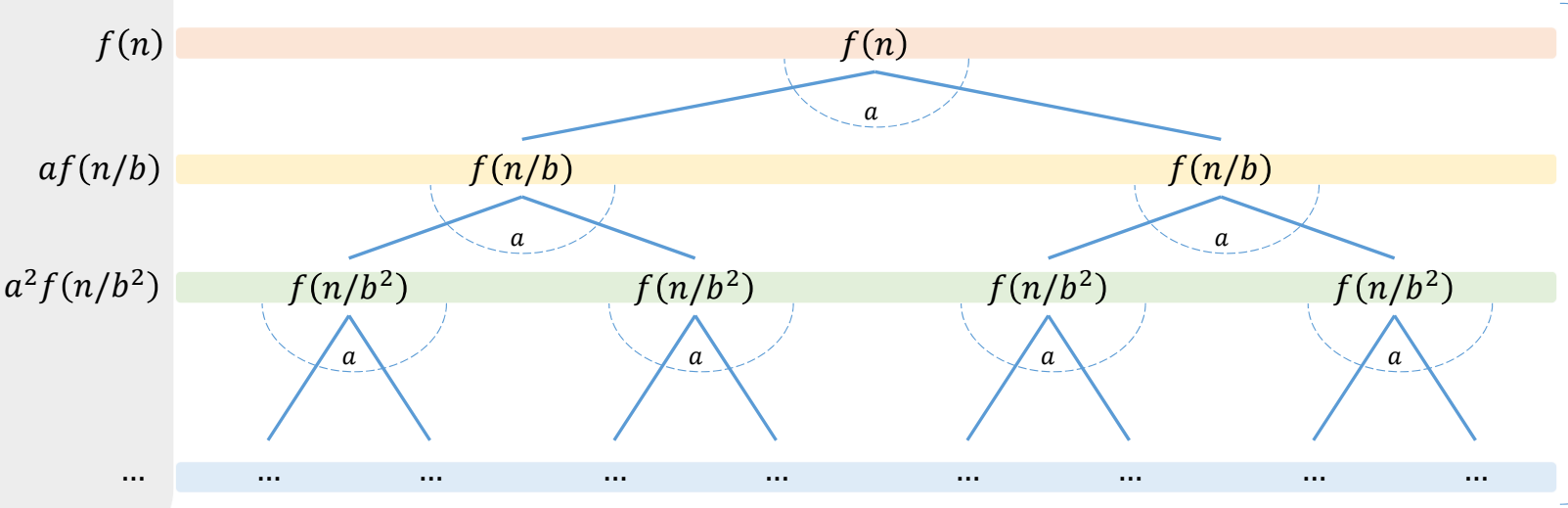
Critical exponent: $d = \log_b a$

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 $O(n^d)$.

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

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

$\log_b n$ levels

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

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

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

...

△

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

△

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

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

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

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

...

△

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

△

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 $\epsilon > 0$.

Proof of the master theorem

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

Splitting/combining:

$\log_b n$ levels

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

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

$b^d = a$

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:

$\log_b n$ levels

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

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

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

...

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

△ $b^d = a$

△

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

Overall cost:

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

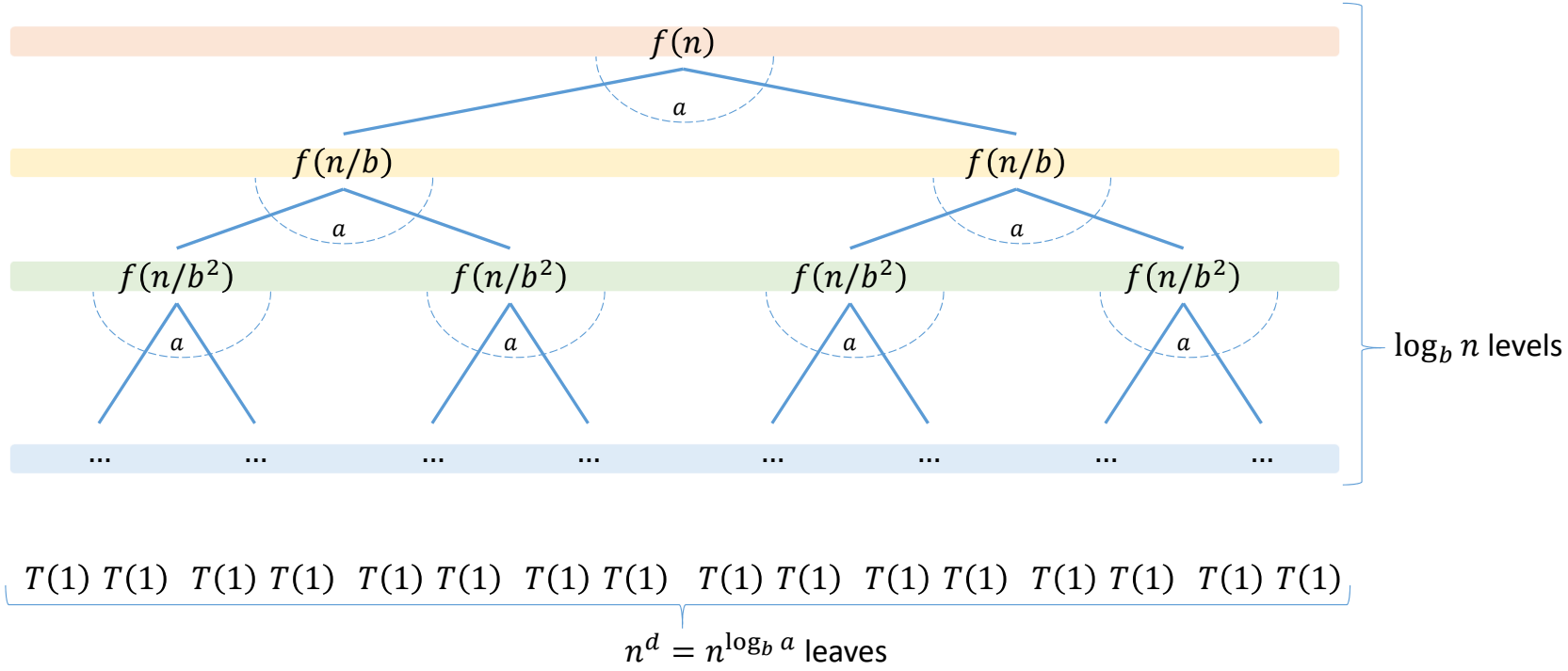
$\log_b n$ terms

Proof of the master theorem

Critical exponent: $d = \log_b a$

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

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

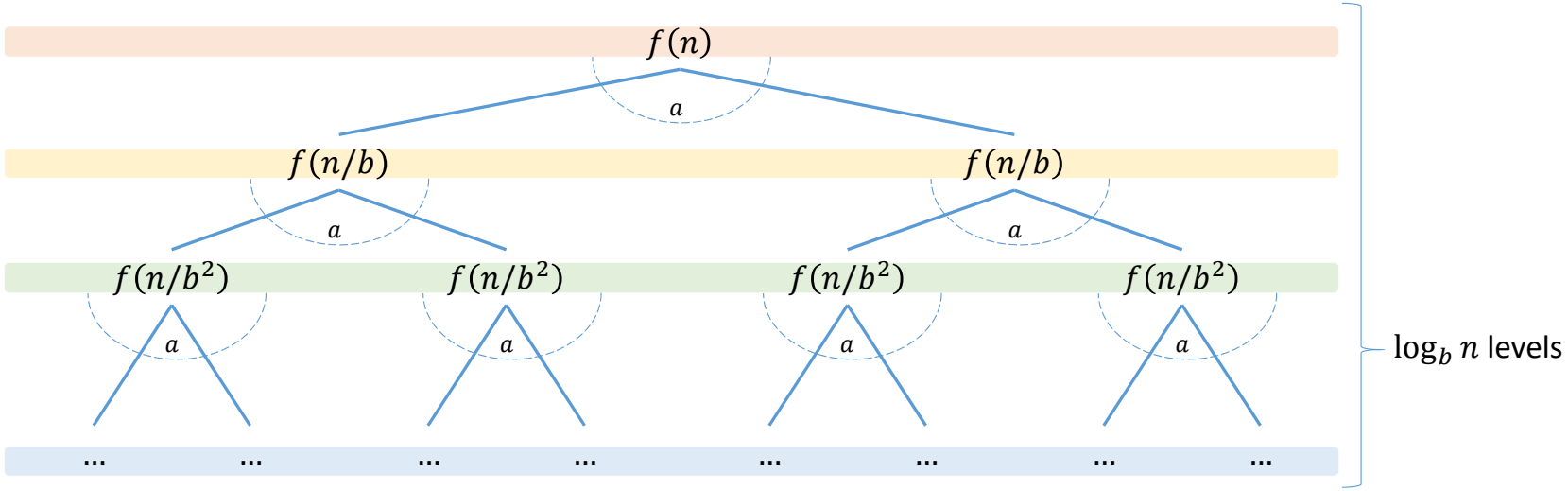


Proof of the master theorem

Critical exponent: $d = \log_b a$

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

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



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

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

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

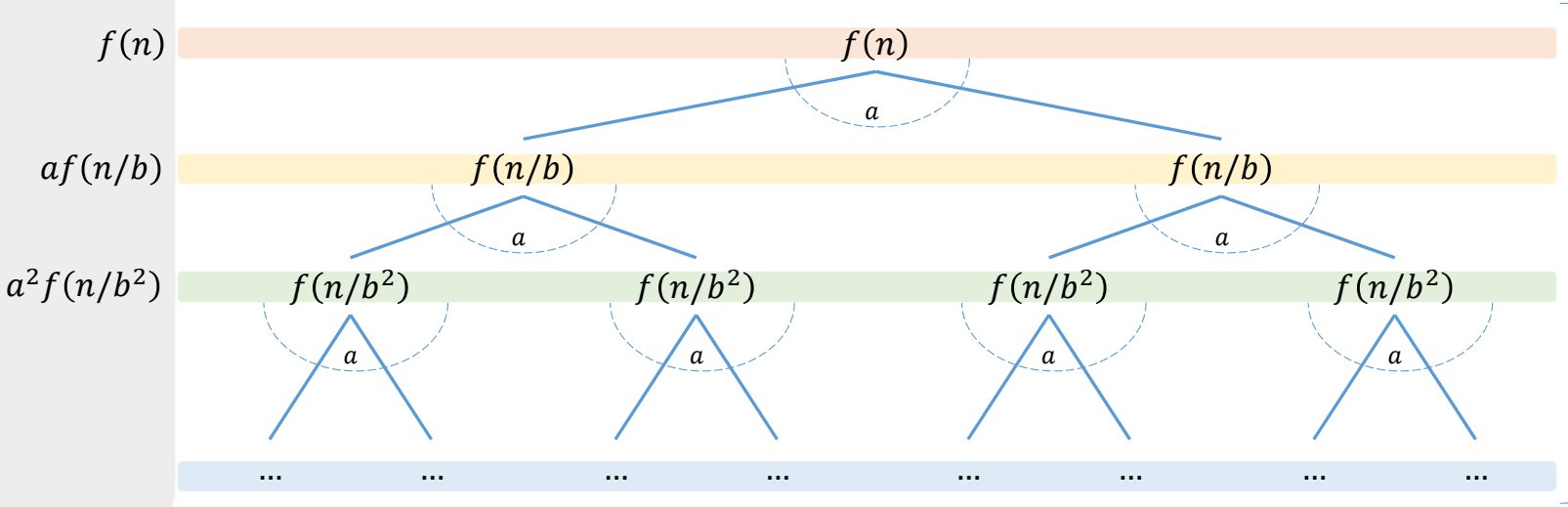
Critical exponent: $d = \log_b a$

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

Proof of the master theorem

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

Splitting/combining:



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

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

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

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

Critical exponent: $d = \log_b a$

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

Proof of the master theorem

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

Splitting/combining:

$\log_b n$ levels

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

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

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

$\vdots$

...

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

Critical exponent: $d = \log_b a$

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

Proof of the master theorem

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

Splitting/combining:

$\log_b n$ levels

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

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

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

$\vdots$

$$\ldots$$

$$\log^k n + \log^k \frac{n}{b} + \log^k \frac{n}{b^2} + \cdots = (\log n)^k + (\log n - \log b)^k + (\log n - 2\log b)^k + \cdots = \Theta\left((\log n)^{k+1}\right)$$

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

$\log_b n$ terms

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

Floors and ceilings

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

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

- **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
  - Wing-Kin Sung