CH08-320201
Algorithms and Data Structures

Lecture 4 — 13 Feb 2018

Prof. Dr. Michael Sedlmair

Jacobs University

Spring 2018

1.5 First general analysis task: Solve recurrences

Solving Recurrences

- Merge Sort analysis required us to solve the recurrence:

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

- A recurrence (or recurrence relation) is an equation that recursively defines a sequence (given an initial term).
- How can we generally solve recurrences?

Three methods

- Substitution method
- Recursion tree
- Master method

Substitution method

The substitution method is based on some intuition. It executes the following steps:

- Guess the form of the solution.
- Verify by induction.
- Solve for constants.

Example

- Consider recurrence $T(n) = 4T(n/2) + n$ with base case $T(1) = \Theta(1)$.
- Prove O and Ω separately.
- Guess $O(n^3)$.
- Verify by induction:
  1. Check base case ($n = 1$).
  2. Assuming $T(k) \leq ck^3$ for $k < n$ show $T(n) \leq cn^3$.

Example

Base case:

$T(n) = \Theta(1)$ for all $n < n_0$, where n_0 is a suitable constant.

For $1 \leq n < n_0$, we have $\Theta(1) \leq cn^3$, if we pick c big enough.

Example

Induction step:

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^3 + n \\
&= (c/2)n^3 + n \\
&= cn^3 - ((c/2)n^3 - n) \leftarrow \textit{desired} - \textit{residual} \\
&\le cn^3 \leftarrow \textit{desired}
\end{aligned}
$$

whenever $\underbrace{(c/2)n^3 - n}_{\textit{residual}} \ge 0$, for example,
if $c \ge 2$ and $n \ge 1$.

Example

- Was our guess a good one?
- Was it tight enough?
- Make a new guess: $T(n) = O(n^2)$.
- Try to prove by induction.
    1. Base step: as before.
    2. Induction step: Assuming $T(k) \leq ck^2$ for $k < n$, show $T(n) \leq cn^2$.

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leq 4c(n/2)^2 + n \\
&= cn^2 + n \\
&= cn^2 - (-n) \quad [\text{ desired} - \text{residual }] \\
&\leq cn^2 \quad \text{for } \textbf{\textit{no}} \text{ choice of } c > 0. \text{ Lose!}
\end{aligned}
$$

Example

- Idea: Adjust hypothesis by subtracting a lower-order term.
- Induction step:
  Assuming $T(k) \leq c_1k^2 - c_2k$ for $k < n$
  show $T(n) \leq c_1n^2 - c_2n$.

$$\begin{aligned}
T(n) &= 4T(n/2) + n \\
&= 4(c_1(n/2)^2 - c_2(n/2)) + n \\
&= c_1n^2 - 2c_2n + n \\
&= c_1n^2 - c_2n - (c_2n - n) \\
&\leq c_1n^2 - c_2n \quad \text{if } c_2 \geq 1.
\end{aligned}$$

Example

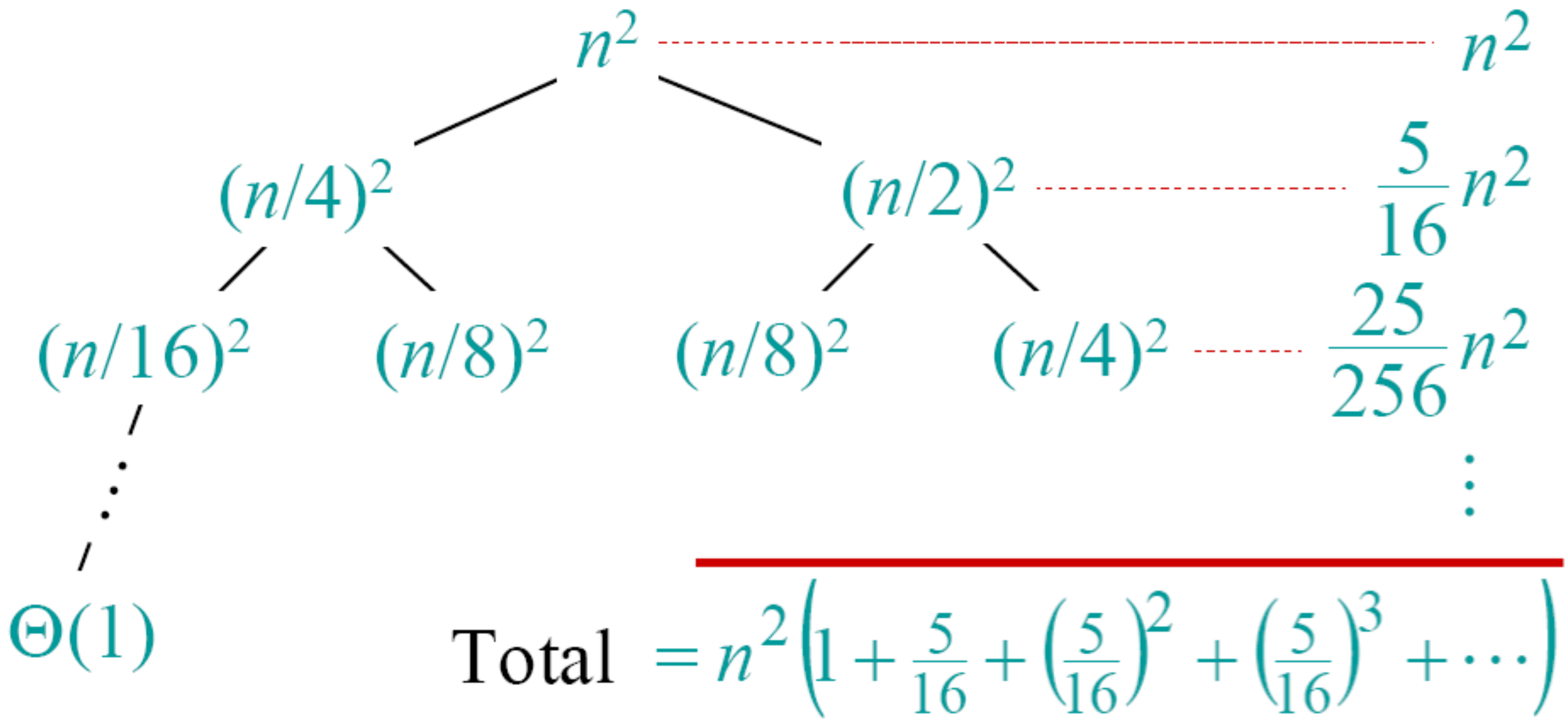
- Finally, solve for constants:
  - Pick c_1 large enough to handle the base case.
  - Pick c_2 according to the induction proof (>1).

Recursion tree

- For the Merge Sort analysis, we used a recursion tree.
- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- This does not necessarily lead to a reliable solution.
- However, the recursion-tree method promotes intuition.
- It is good for generating guesses for the substitution method.

Example

Consider recurrence $T(n) = T(n/4) + T(n/2)+ n^2$ with base case $T(1) = \Theta(1)$.



Example

Considering the geometric series

$$1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x} \quad \text{for } x \neq 1$$

$$1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1$$

we get $T(n) = \Theta(n^2)$

Master method

The master method applies to recurrences of the form

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

where $a \geq 1$, $b > 1$, and f is asymptotically positive.

It distinguishes 3 common cases by comparing

$$f(n) \text{ with } n^{\log_b a}$$

Master method

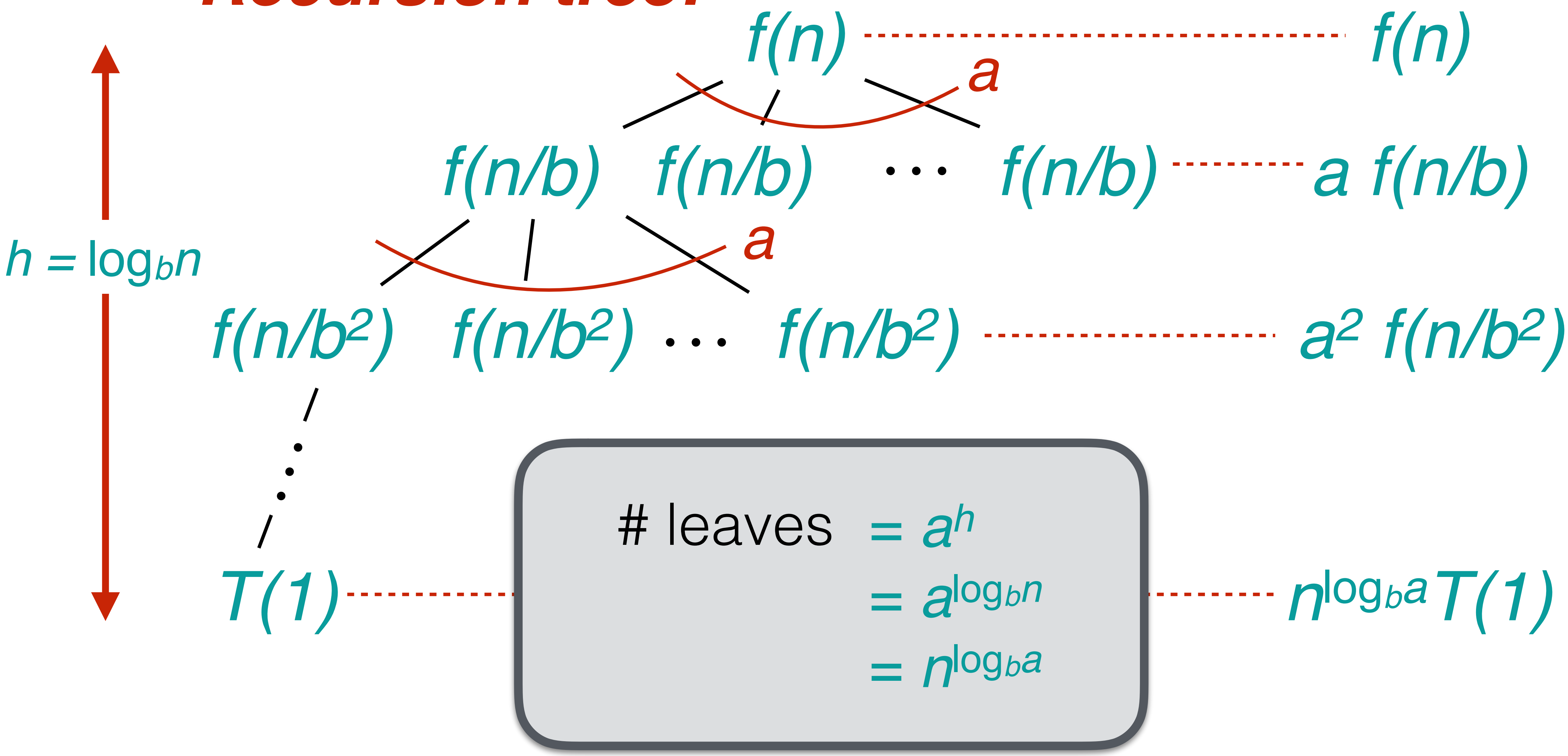
Recurrence: $T(n) = aT(n/b) + f(n)$

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ and $a\, f(n/b) \leq c\, f(n)$ for some constant $c < 1$, then $T(n) = \Theta(f(n))$.

Idea of master theorem

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

Recursion tree:



Idea of master theorem

Recursion tree:

$f(n)$ .... $f(n)$

a

$f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b)$ .... $a\, f(n/b)$

a

$h = \log_b n$

$f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2)$ .... $a^2\, f(n/b^2)$

$\vdots$

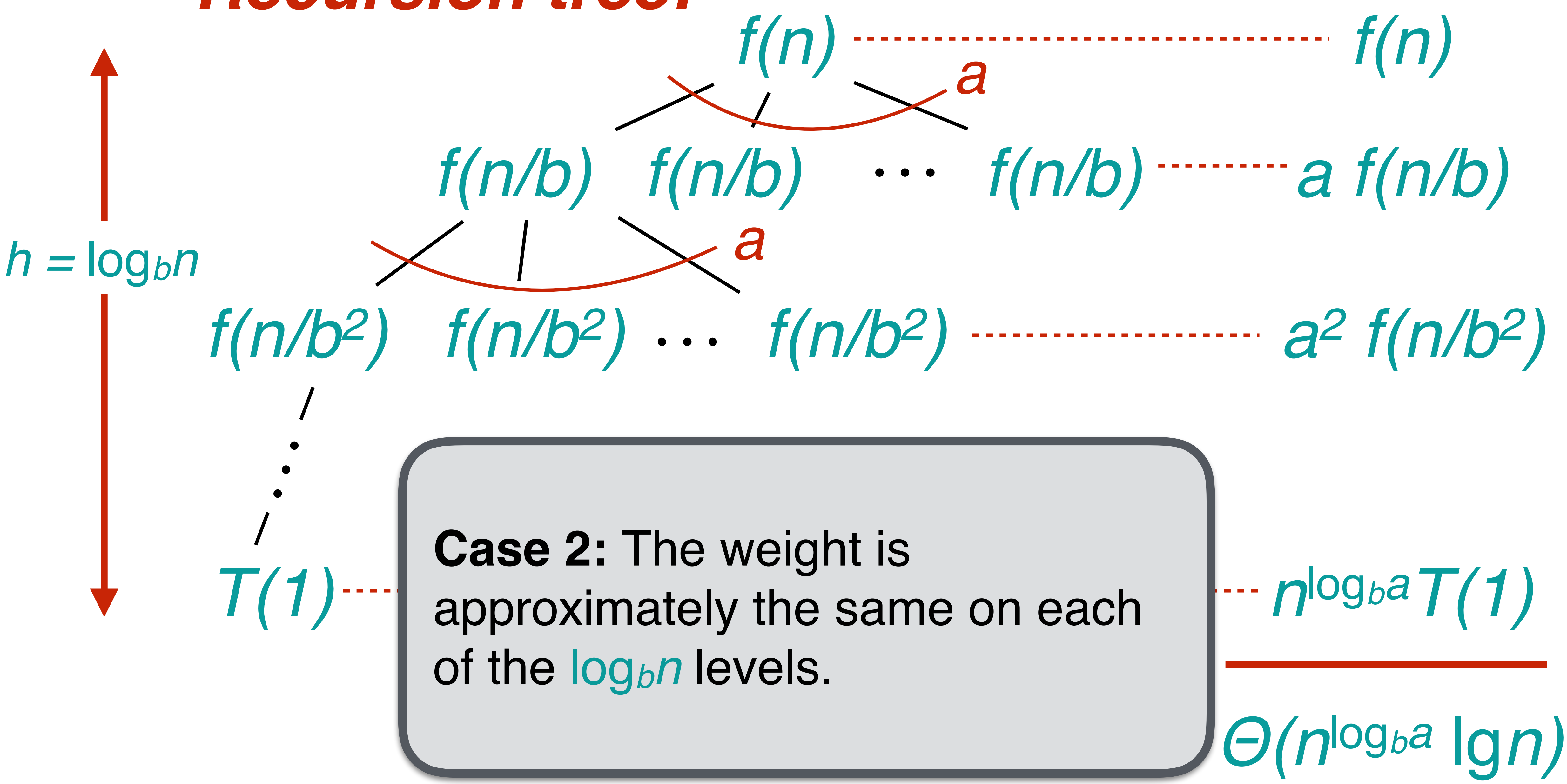
$T(1)$ .... $n^{\log_b a}\, T(1)$

$\Theta(n^{\log_b a})$

Case 1: The weight **increases** geometrically from the root to the leaves. The **leaves** hold a constant fraction of the total weight.

Idea of master theorem

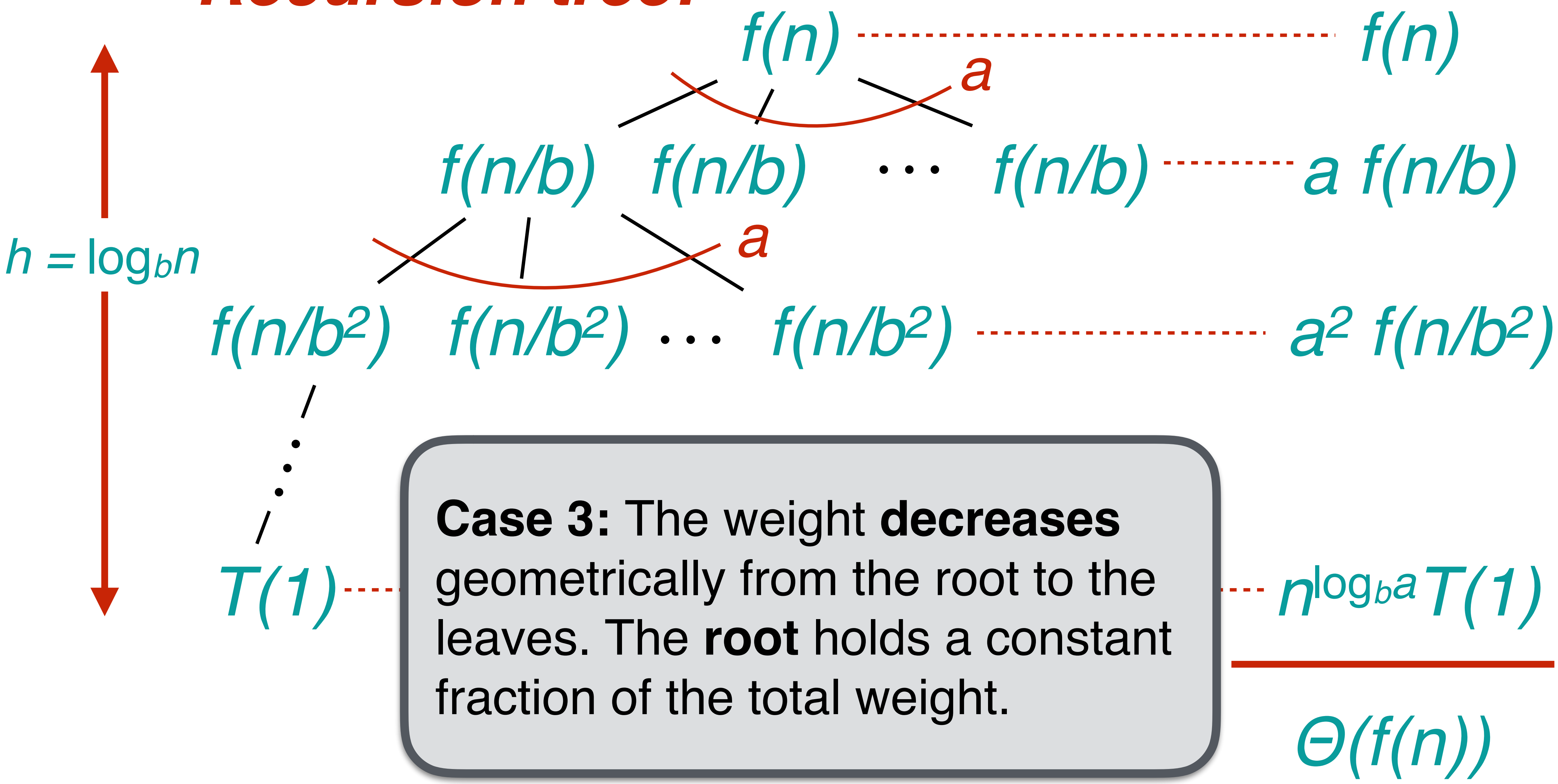
Recursion tree:



Case 2: The weight is approximately the same on each of the $\log_b n$ levels.

Idea of master theorem

Recursion tree:



Case 3: The weight **decreases** geometrically from the root to the leaves. The **root** holds a constant fraction of the total weight.

Example 1

$T(n) = 4T(n/2) + n$
$a = 4,\ b = 2$

$n^{\log_b a} = n^2$
$f(n) = n$

Case 1: $f(n) = O\,(n^{2-\varepsilon})$ for $\varepsilon = 1$
Thus, $T(n) = \Theta(n^2)$.

Example 2

$T(n) = 4T(n/2) + n^2$
$a = 4,\ b = 2$

$n^{\log_b a} = n^2$
$f(n) = n^2$

Case 2: $f(n) = \Theta(n^2)$,
Thus, $T(n) = \Theta(n^2 \lg n)$.

Example 3

$T(n) = 4T(n/2) + n^3$
$a = 4,\ b = 2$

$n^{\log_b a} = n^2$
$f(n) = n^3$

Case 3: $f(n) = \Omega\,(n^{2+\varepsilon})$ *for* $\varepsilon = 1$
and $4(n/2)^3 \leq cn^3$ for $c = 1/2$ (regulation condition)
Thus, $T(n) = \Theta(n^3)$.

Example 4

$T(n) = 4T(n/2) + n^2/\lg n$
$a = 4,\ b = 2$

$n^{\log_b a} = n^2$
$f(n) = n^2/\lg n$

Master method does not apply
(for every constant $\varepsilon > 0$, we have $n^{\varepsilon} = \omega(\lg n)$)