

Data Structures and Algorithms

CMPSC 465

LECTURE 9

Solving recurrences

- Master theorem

Adam Smith

Review questions

- Guess the solution to the recurrence:

$$T(n) = 2T(n/3) + n^{3/2}.$$

(Answer: $\Theta(n^{3/2})$.)

- Draw the recursion tree for this recurrence.

a. What is its height?

(Answer: $h = \log_3 n$.)

b. What is the number of leaves in the tree?

(Answer: $n^{(1/\log 3)}$.)

The 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, that is $f(n) > 0$ for all $n > n_0$.

Three common cases

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

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
 - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^ε factor).

Solution: $T(n) = \Theta(n^{\log_b a})$.

Three common cases

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

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.

- $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^ε factor).

Solution: $T(n) = \Theta(n^{\log_b a})$.

2. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.

- $f(n)$ and $n^{\log_b a}$ grow at similar rates.

Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

Three common cases (cont.)

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

3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.

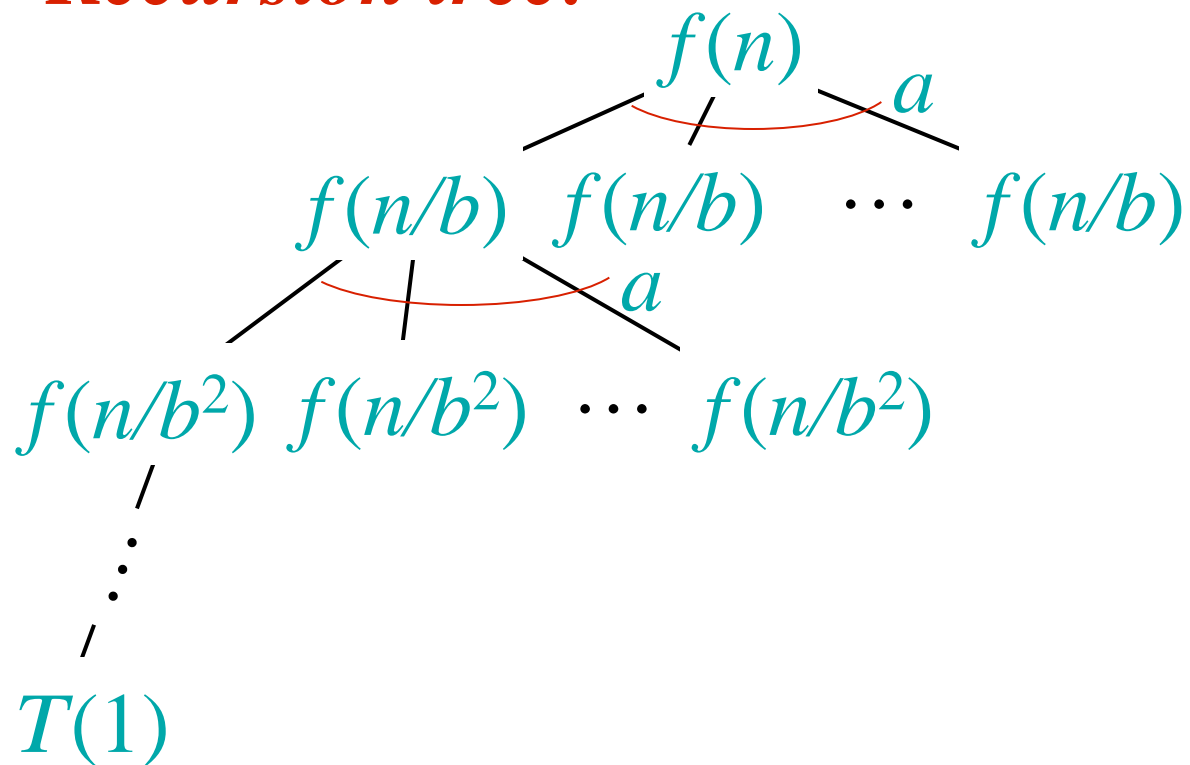
- $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^ε factor),

and $f(n)$ satisfies the **regularity condition** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$.

Solution: $T(n) = \Theta(f(n))$.

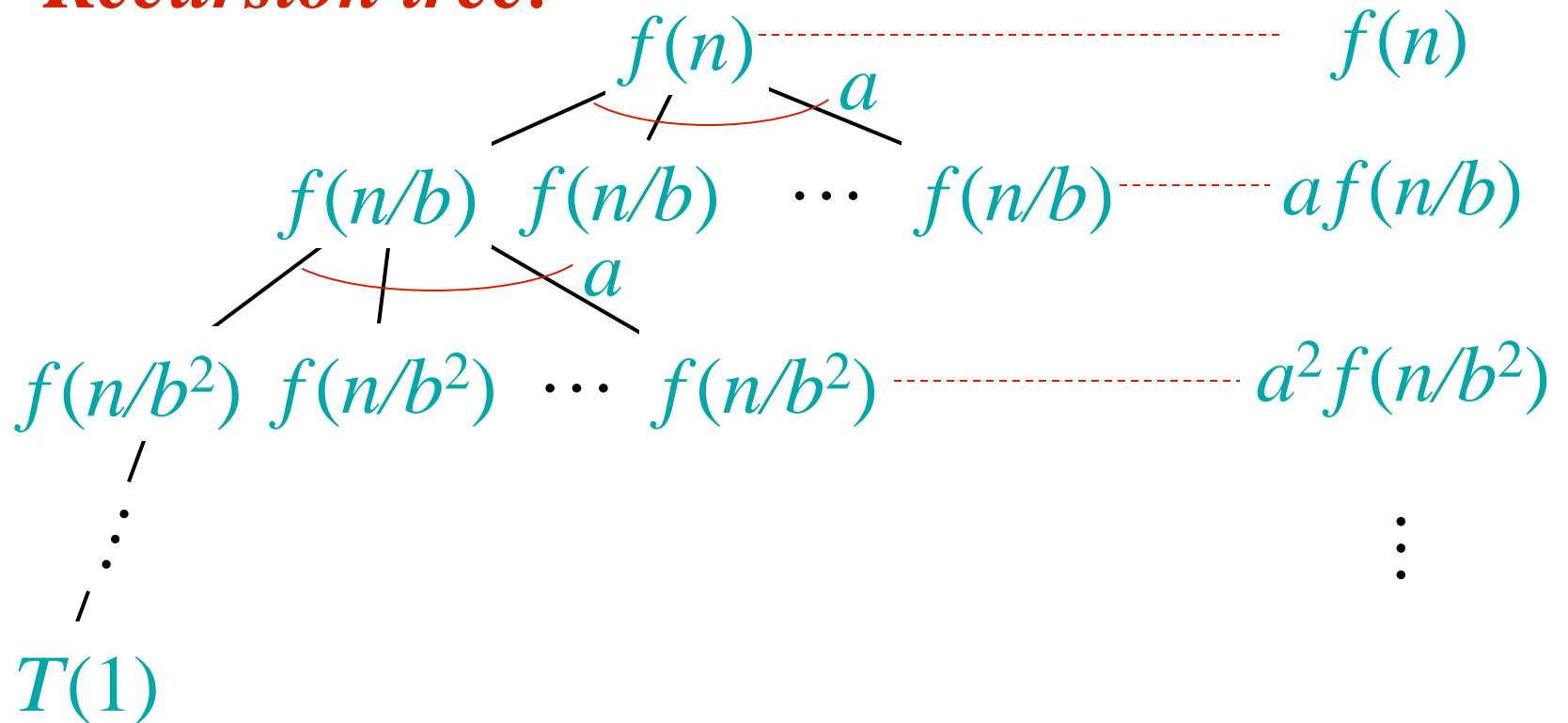
Idea of master theorem

Recursion tree:

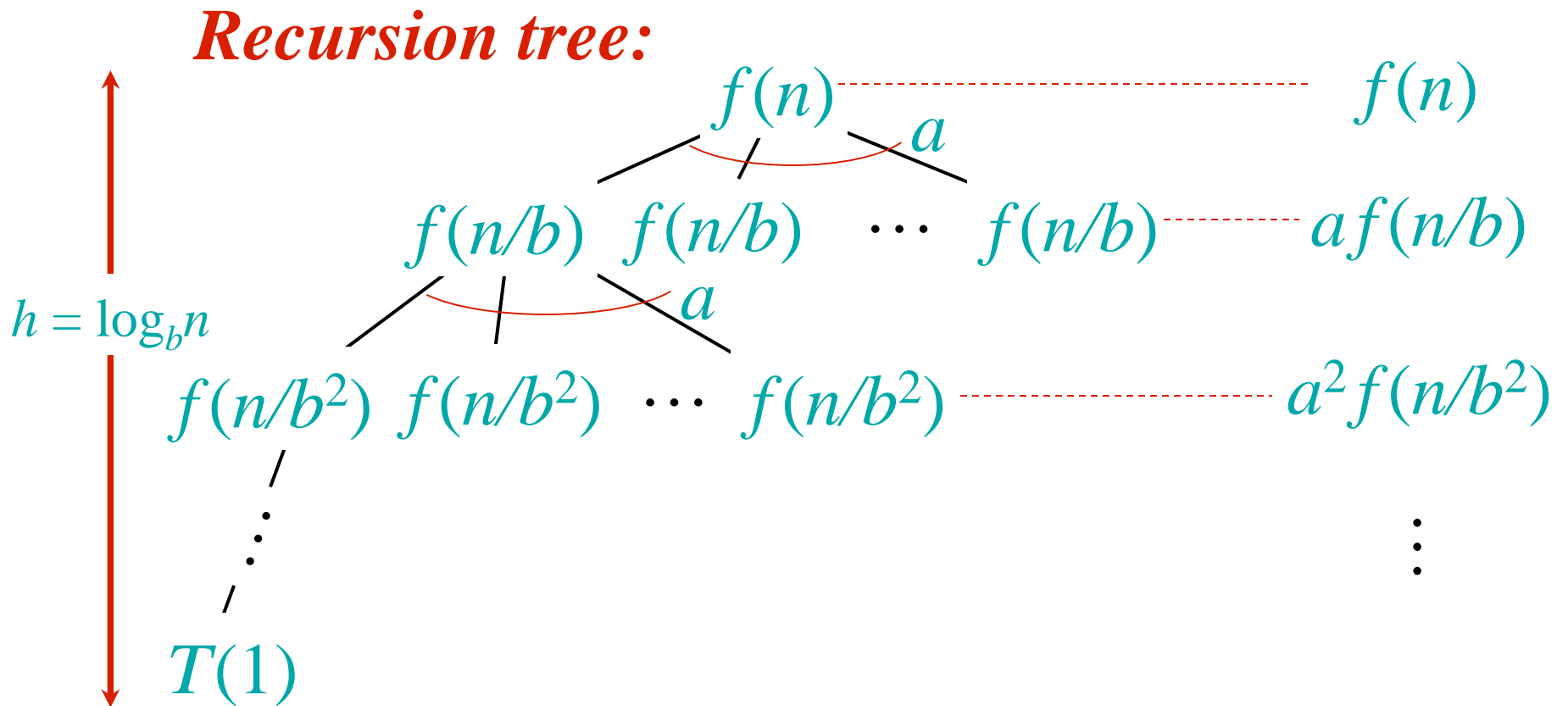


Idea of master theorem

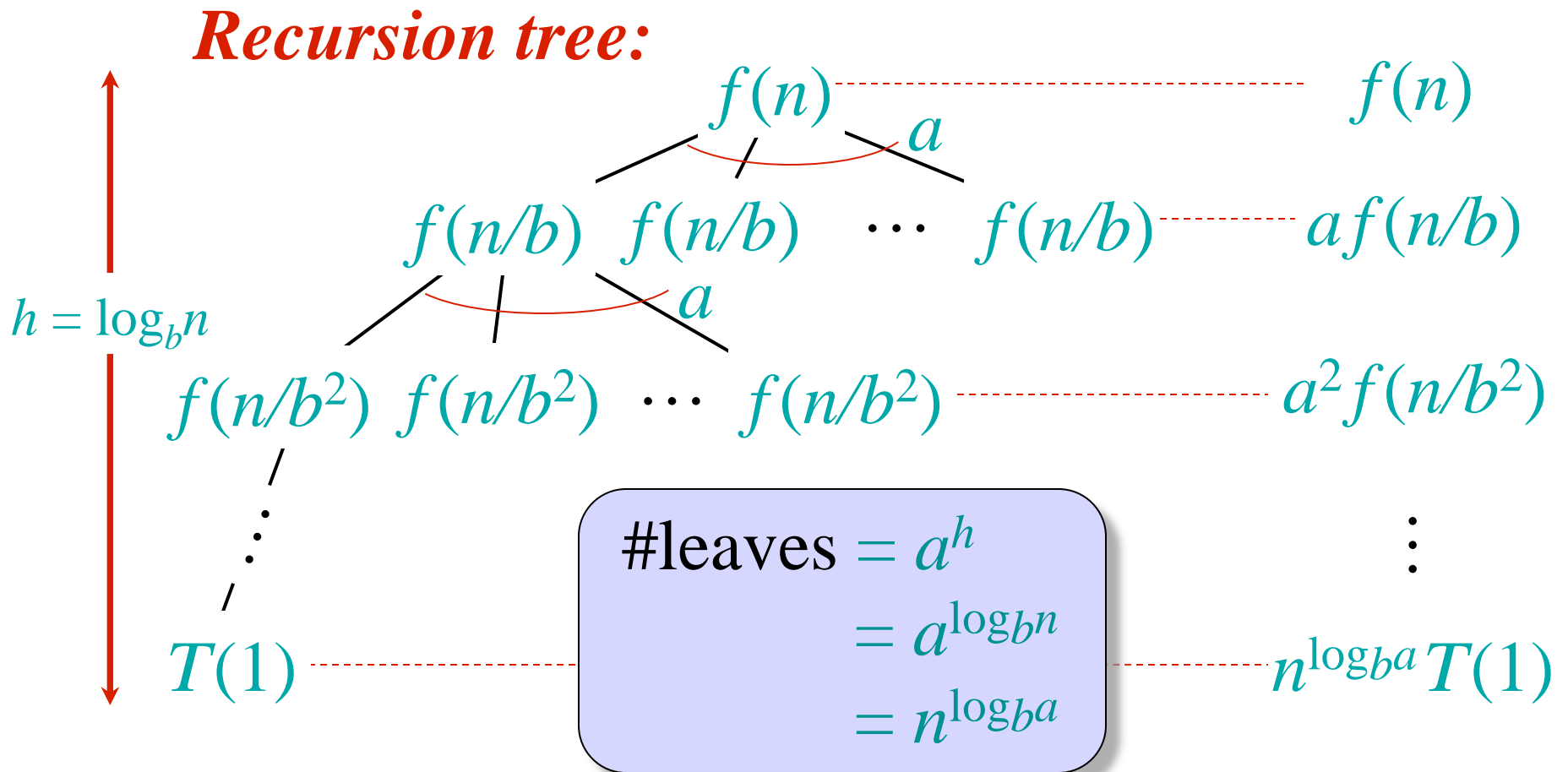
Recursion tree:



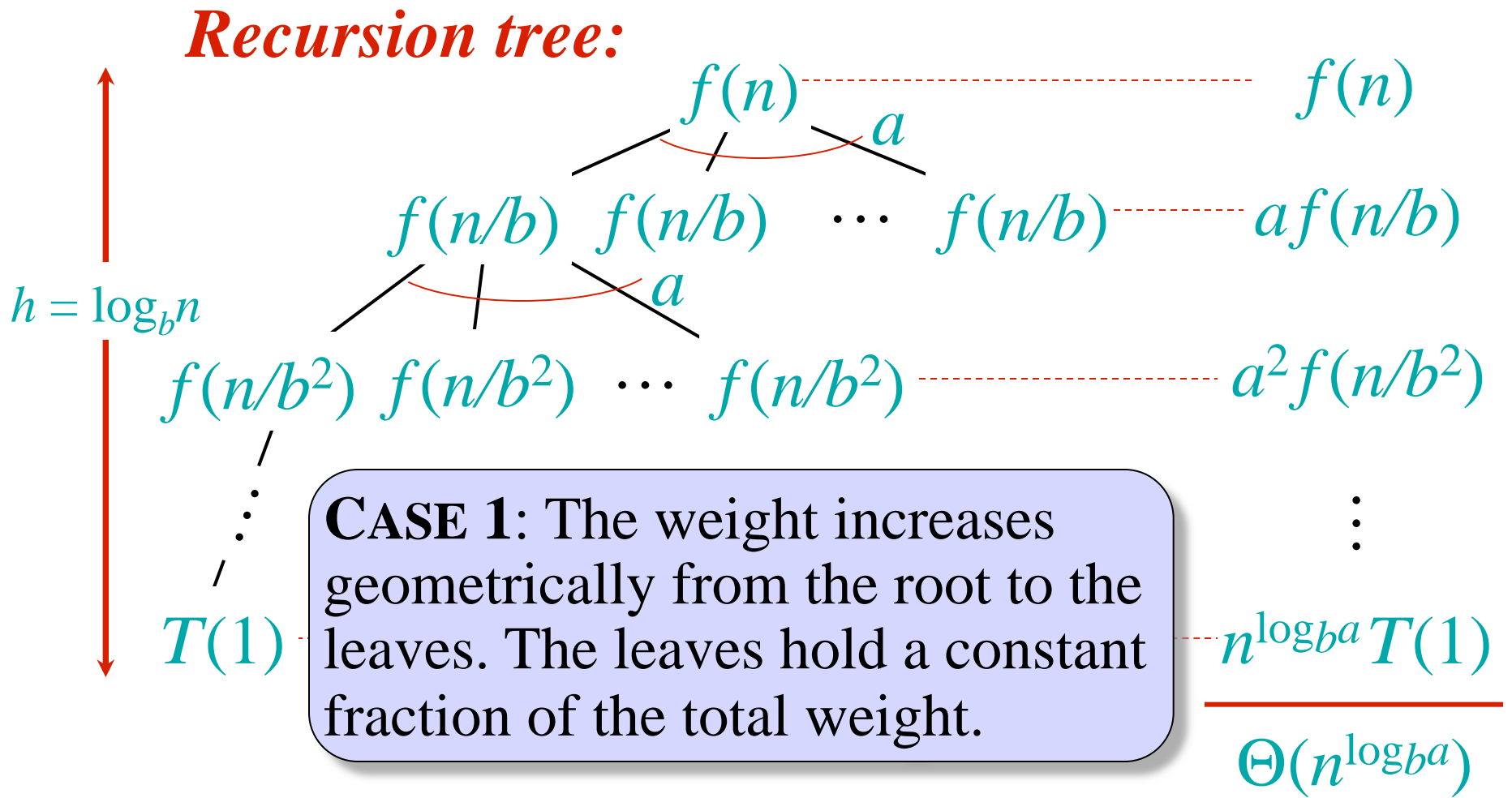
Idea of master theorem



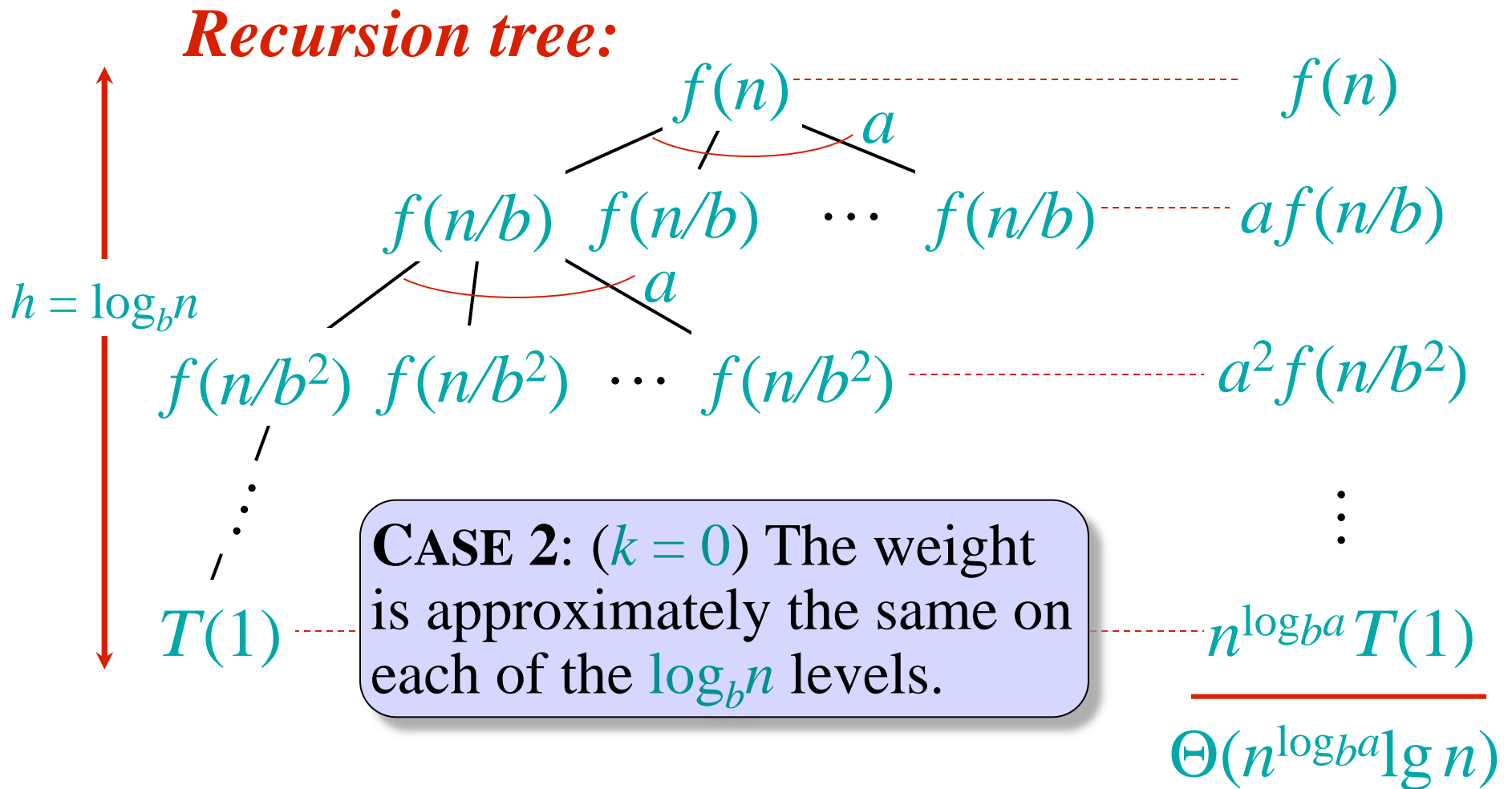
Idea of master theorem



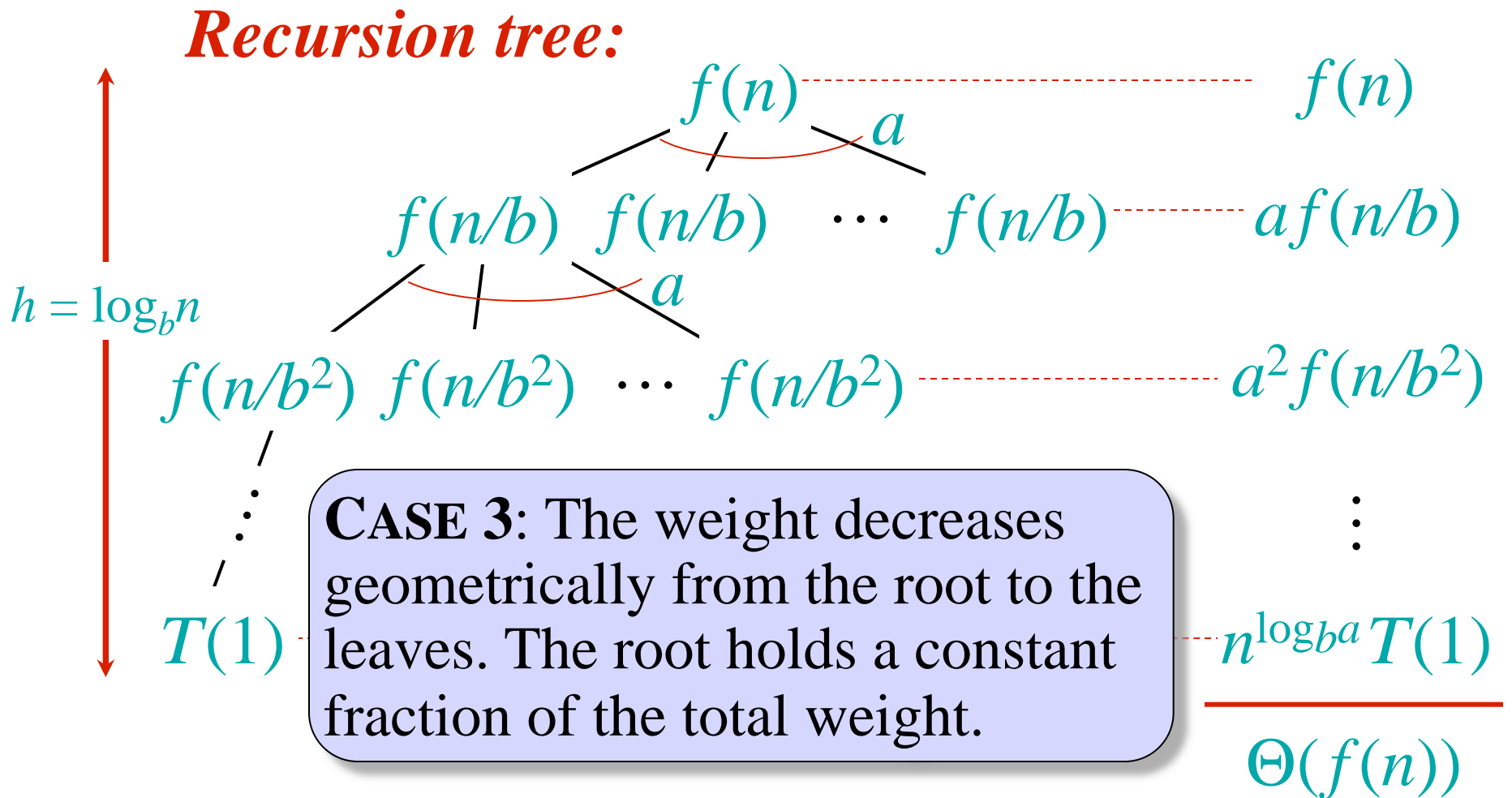
Idea of master theorem



Idea of master theorem



Idea of master theorem



Examples

Ex. $T(n) = 4T(n/2) + n$

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n.$

CASE 1: $f(n) = O(n^{2-\epsilon})$ for $\epsilon = 1.$

$\therefore T(n) = \Theta(n^2).$

Examples

Ex. $T(n) = 4T(n/2) + n$

$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n.$$

CASE 1: $f(n) = O(n^{2-\epsilon})$ for $\epsilon = 1$.

$$\therefore T(n) = \Theta(n^2).$$

Ex. $T(n) = 4T(n/2) + n^2$

$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2.$$

CASE 2: $f(n) = \Theta(n^2 \lg^0 n)$, that is, $k = 0$.

$$\therefore T(n) = \Theta(n^2 \lg n).$$

Examples

Ex. $T(n) = 4T(n/2) + n^3$

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^3.$

CASE 3: $f(n) = \Omega(n^{2 + \epsilon})$ for $\epsilon = 1$

and $4(n/2)^3 \leq cn^3$ (reg. cond.) for $c = 1/2.$

$\therefore T(n) = \Theta(n^3).$

Examples

Ex. $T(n) = 4T(n/2) + n^3$

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^3.$

CASE 3: $f(n) = \Omega(n^{2 + \epsilon})$ for $\epsilon = 1$

and $4(n/2)^3 \leq cn^3$ (reg. cond.) for $c = 1/2.$

$\therefore T(n) = \Theta(n^3).$

Ex. $T(n) = 4T(n/2) + n^2/\lg n$

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2/\lg n.$

Master method does not apply. In particular, for every constant $\epsilon > 0$, we have $n^\epsilon = \omega(\lg n).$