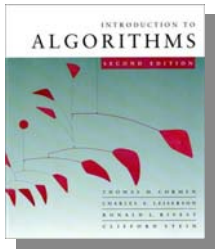
# Design and Analysis of Algorithms

CSE 5311

Lecture 2 Solving Recurrences

Song Jiang, Ph.D.

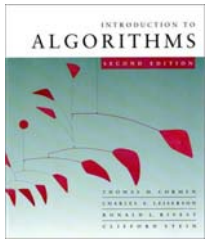
Department of Computer Science and Engineering




# Solving recurrences

- The analysis of merge sort from ***Lecture 1*** required us to solve a recurrence.
- Recurrences are like solving integrals, differential equations, etc.
  - Learn a few tricks.
- ***Lecture 3***: Applications of recurrences to divide-and-conquer algorithms.

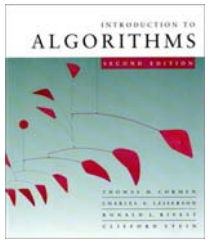
  




# Recurrence for merge sort

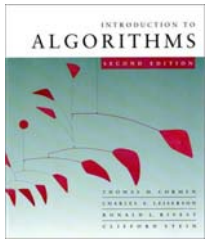
$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

- We shall usually omit stating the base case when $T(n) = \Theta(1)$ for sufficiently small $n$, but only when it has no effect on the asymptotic solution to the recurrence.
- CLRS and Lecture 2 provide several ways to find a good upper bound on $T(n)$.




# Recursion tree

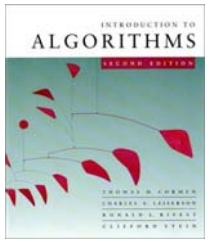
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.




# Recursion tree

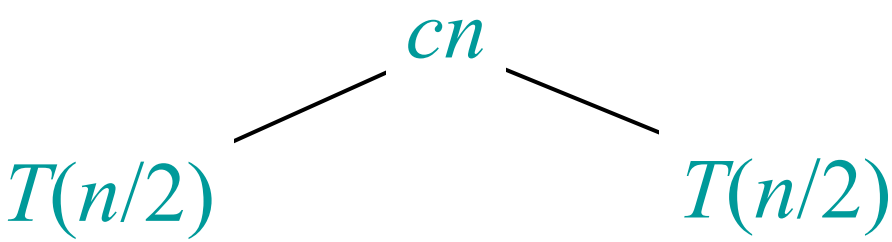
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

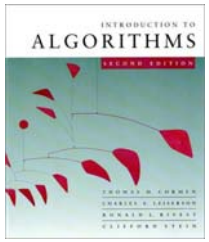
$T(n)$




# Recursion tree

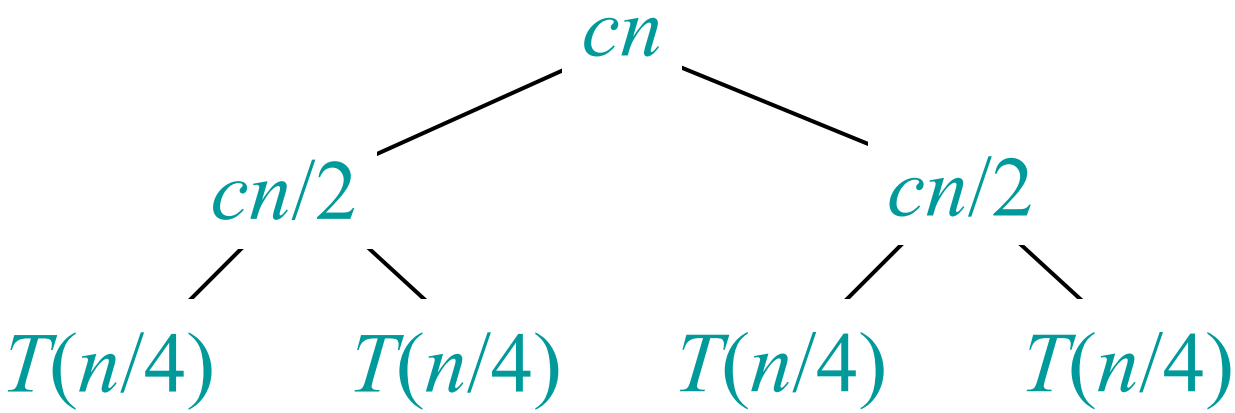
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

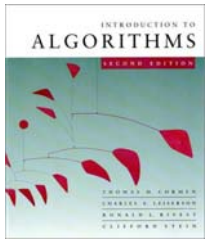





# Recursion tree

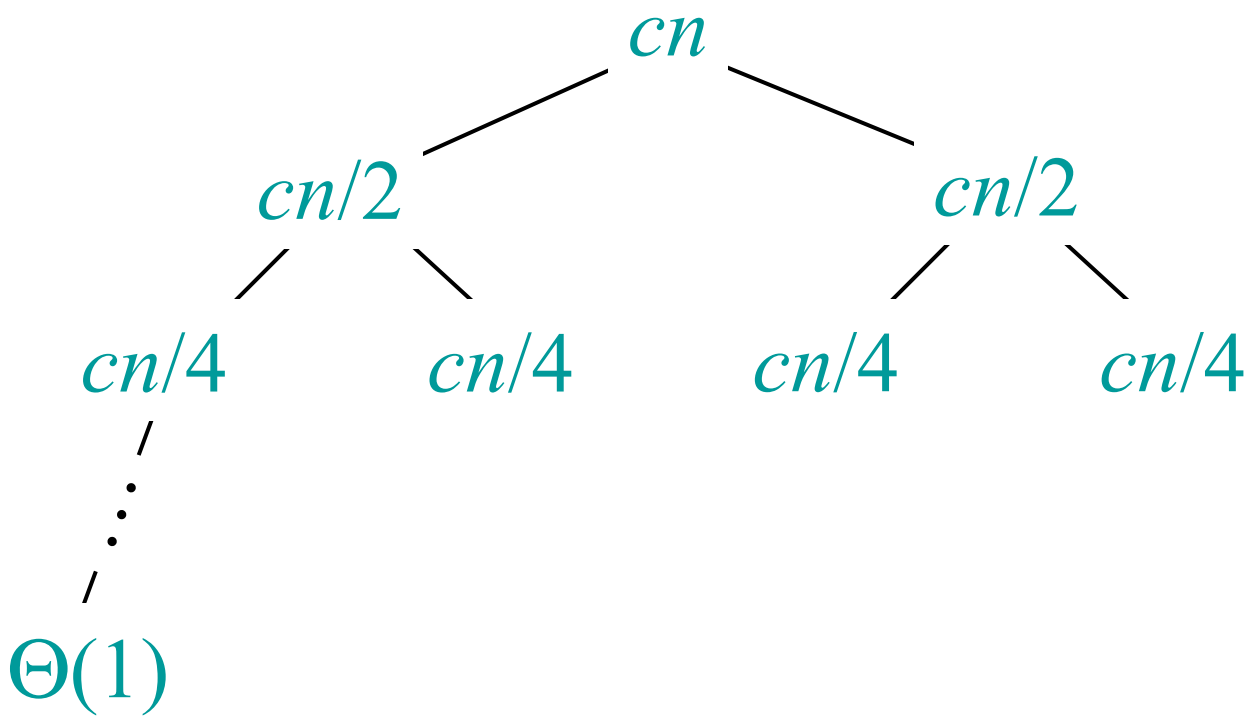
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

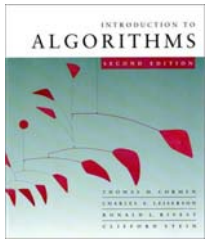





# Recursion tree

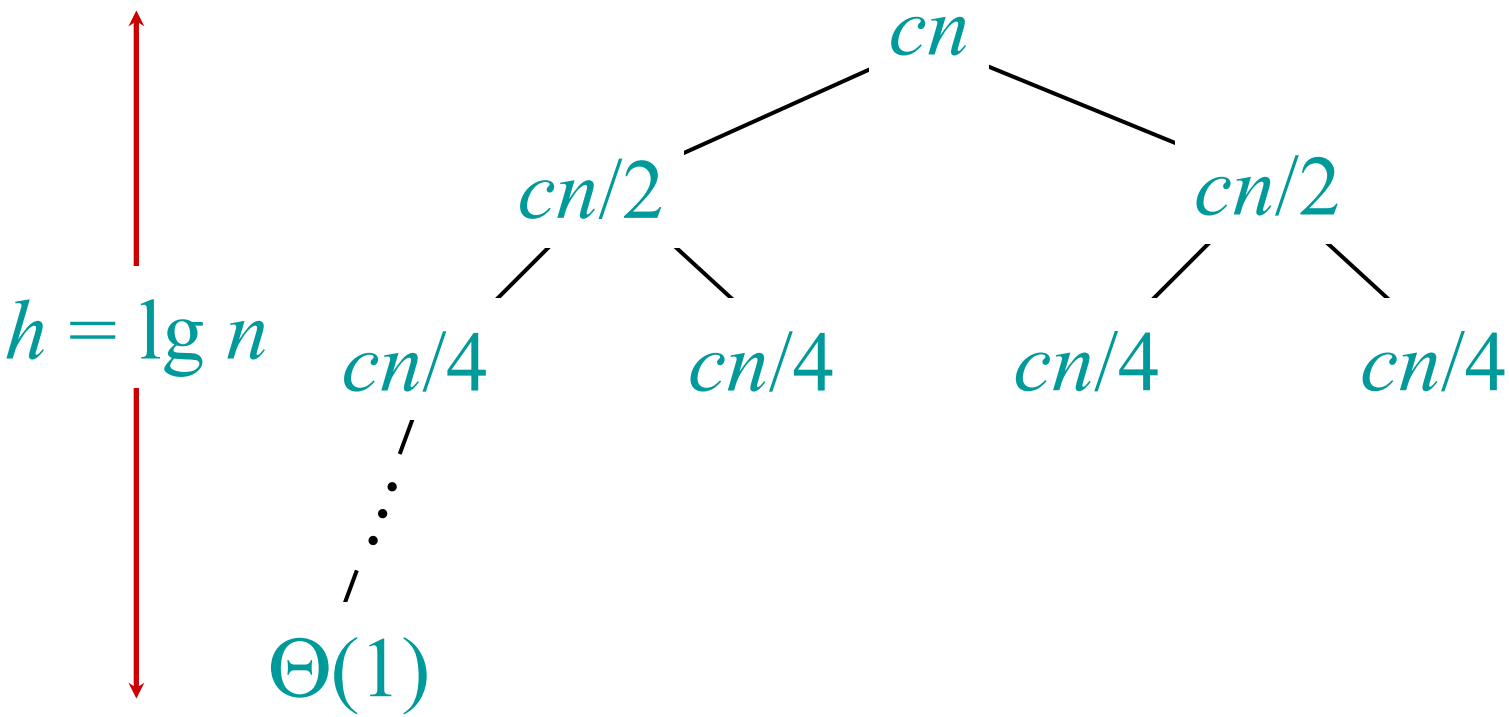
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

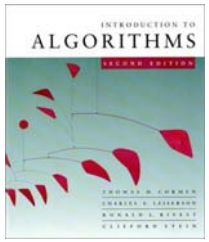





# Recursion tree

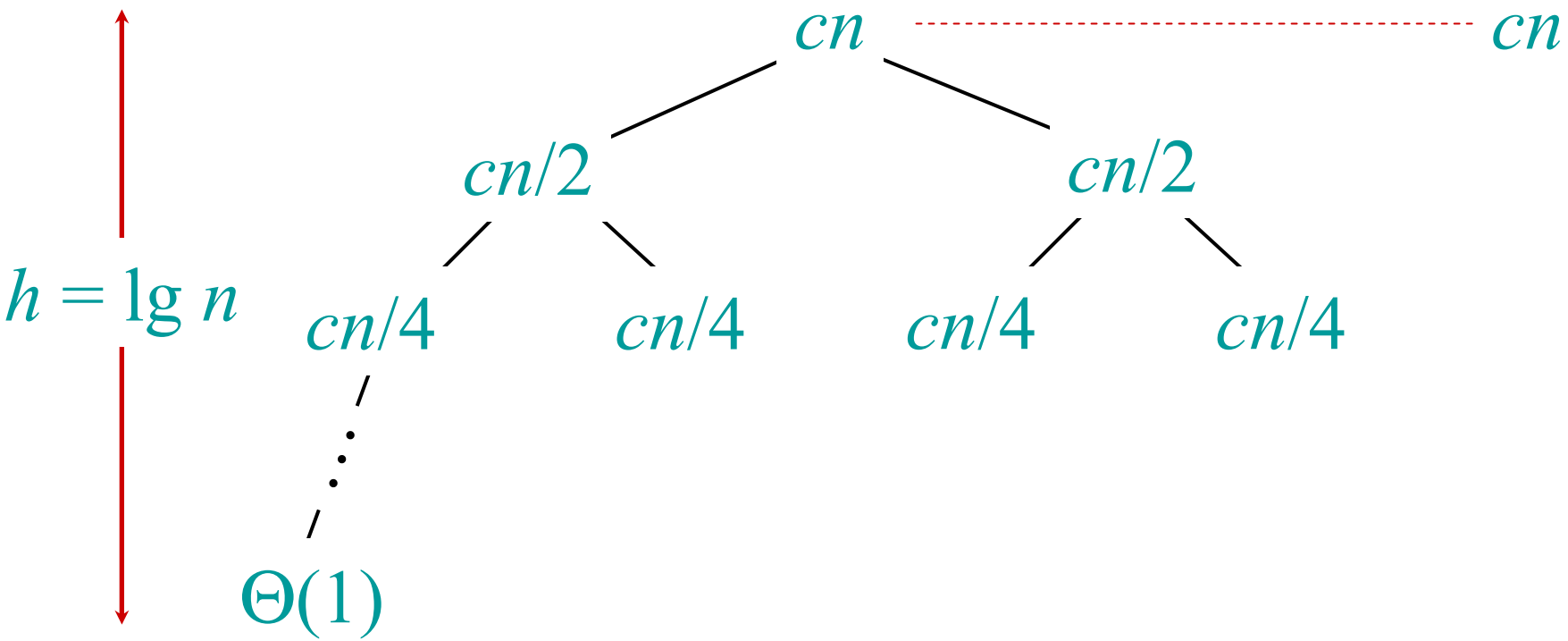
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

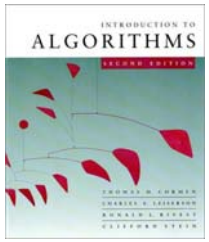





# Recursion tree

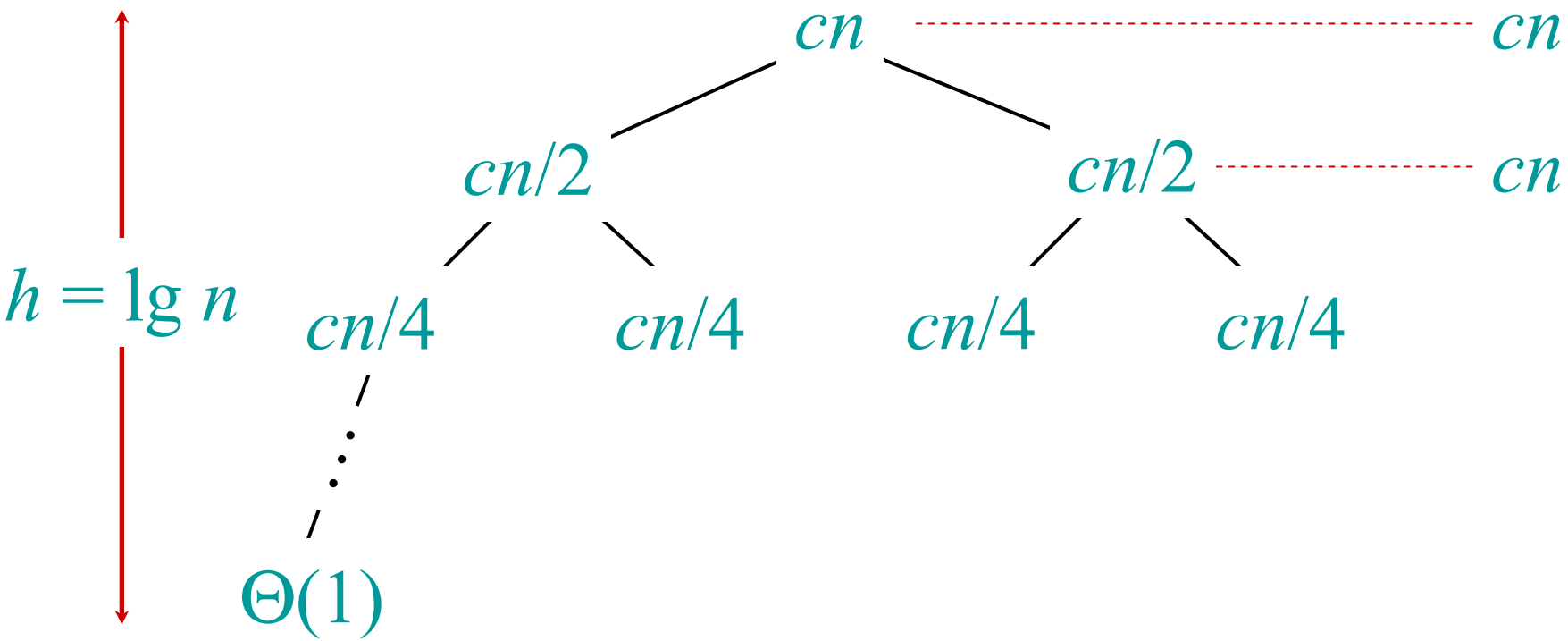
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

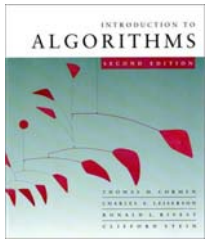





# Recursion tree

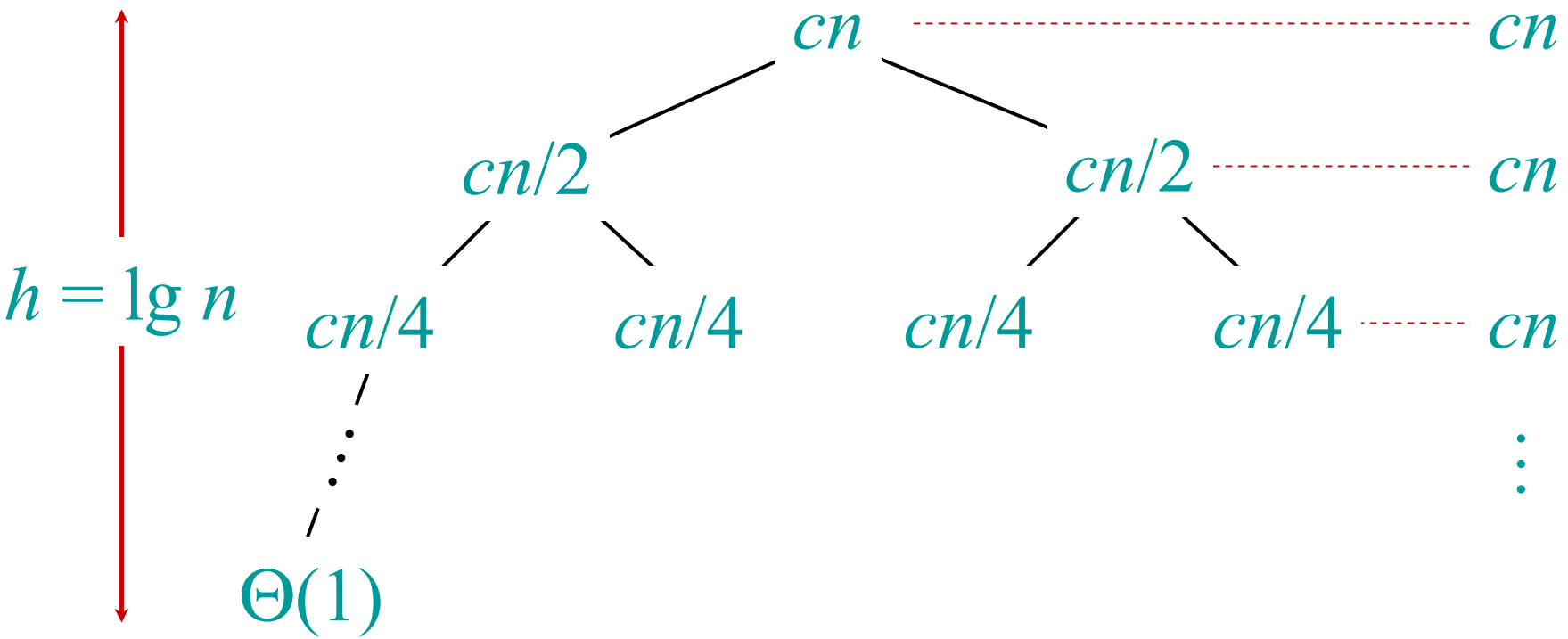
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

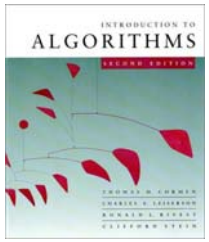





# Recursion tree

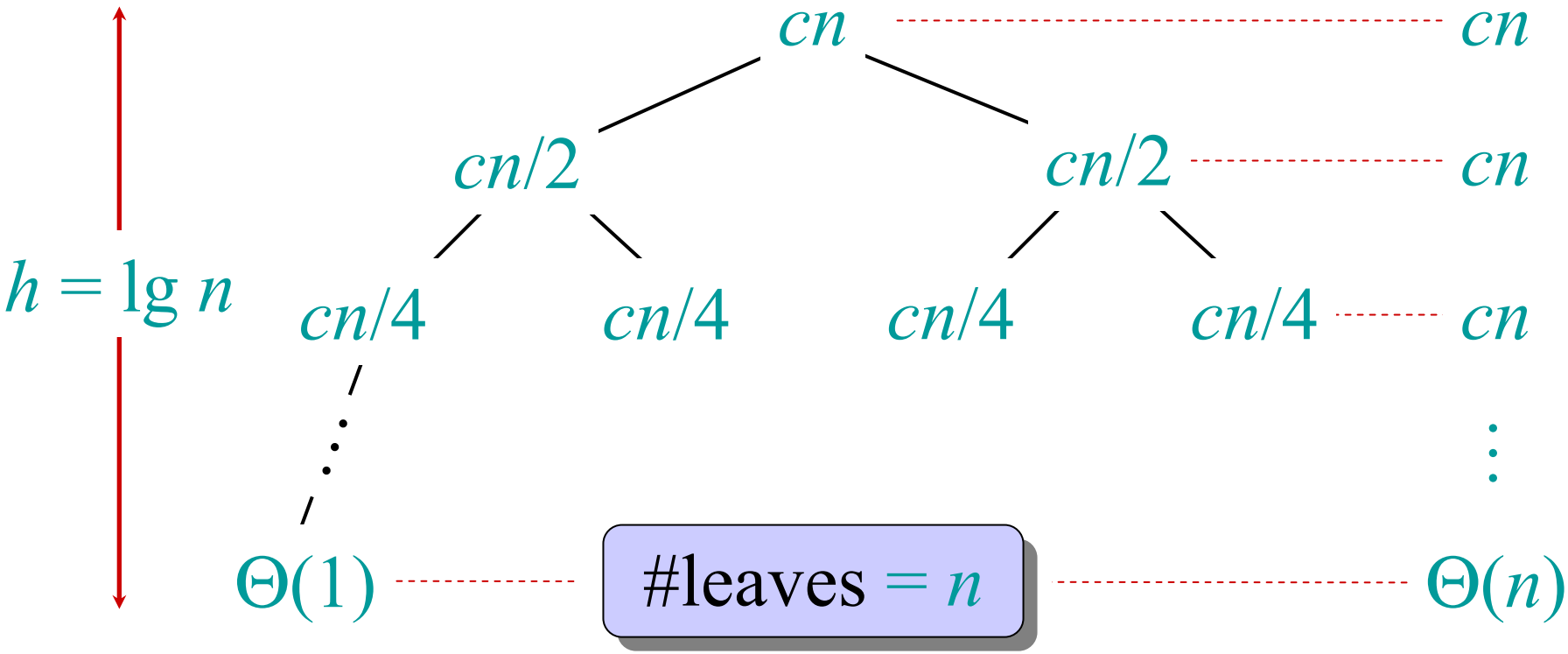
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

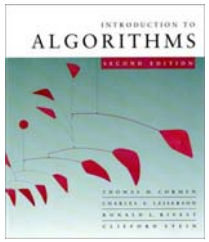





# Recursion tree

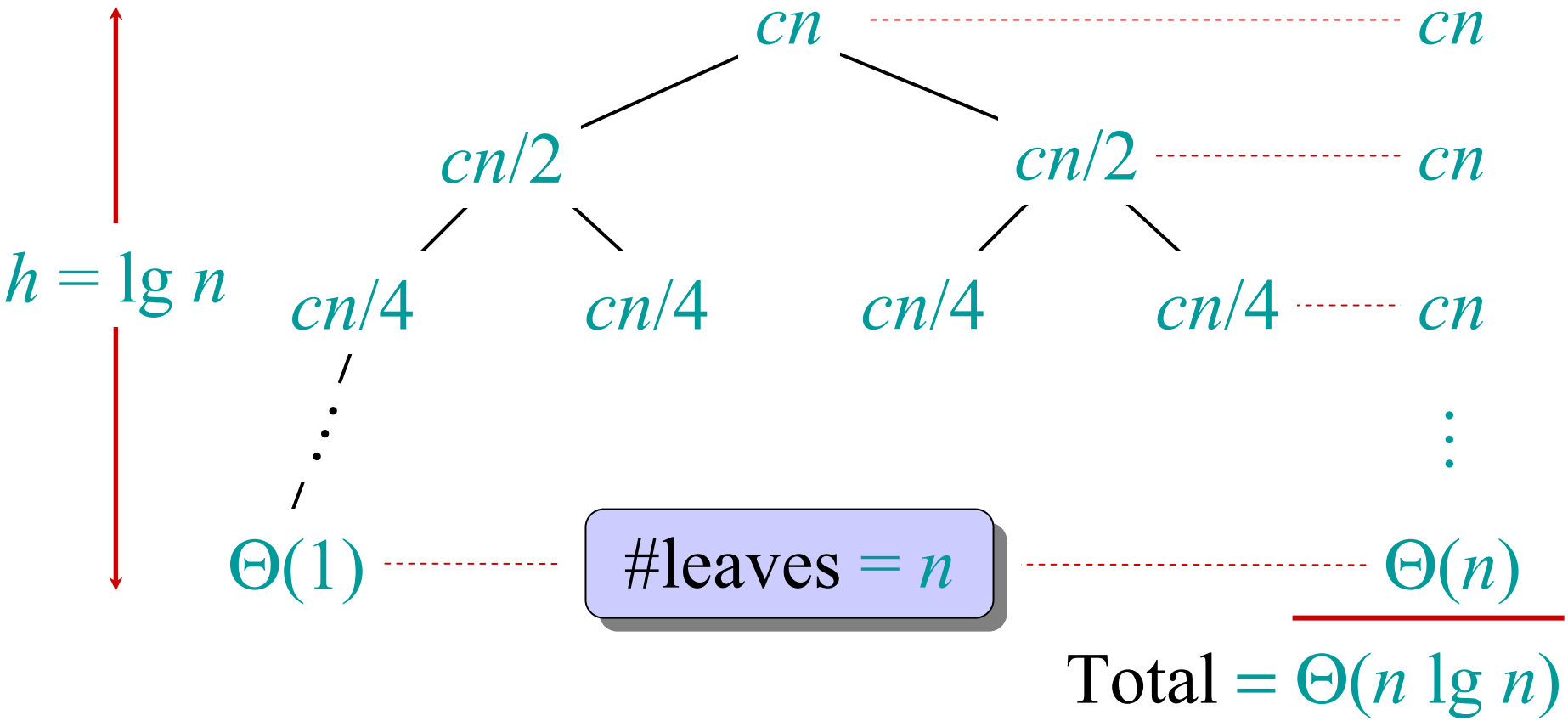
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

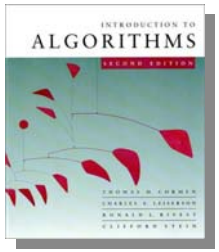





# Recursion tree

Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

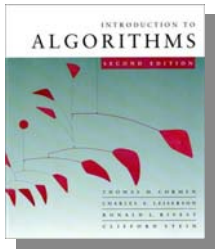





# Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion-tree method can be unreliable, just like any method that uses ellipses (…).
- The recursion-tree method promotes intuition, however.
- The recursion tree method is good for generating guesses for the substitution method.

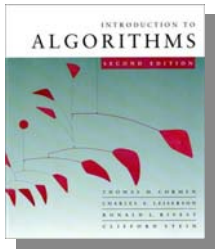
  




# Example of recursion tree

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

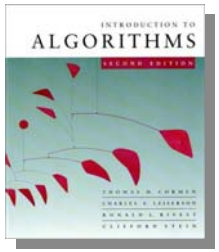



# Example of recursion tree

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

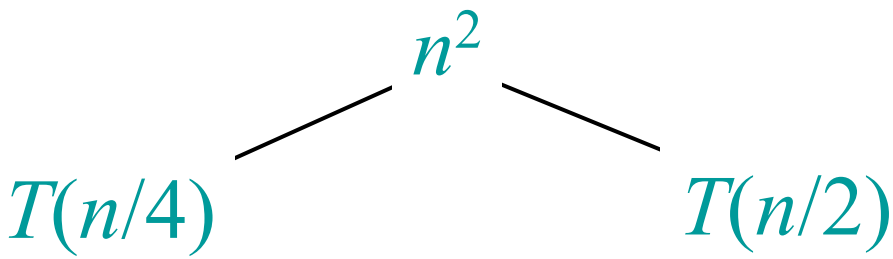
$T(n)$

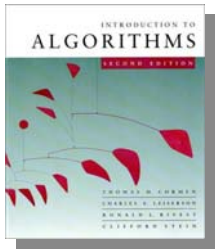



# Example of recursion tree

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

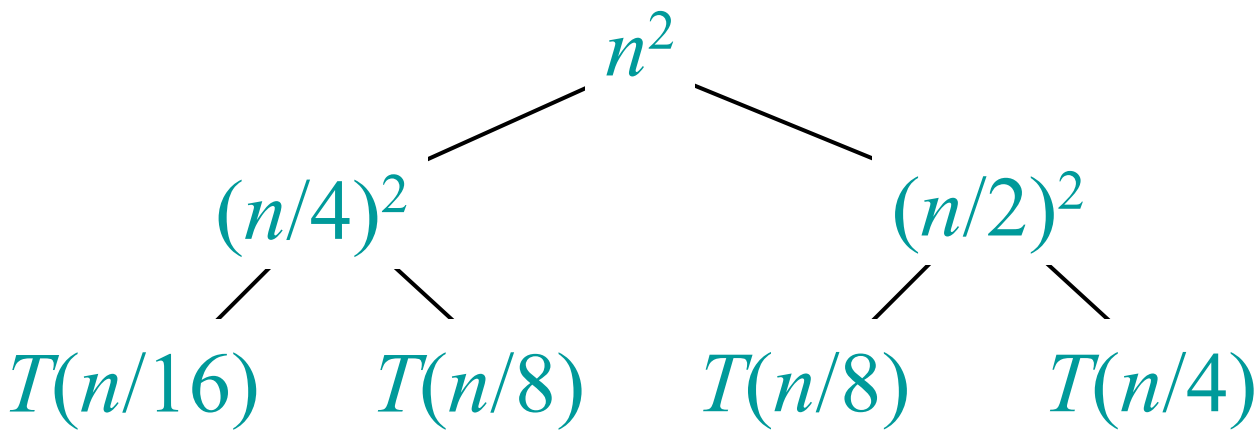


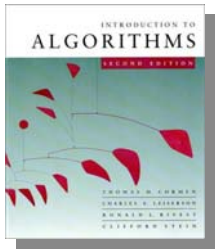

  




# Example of recursion tree

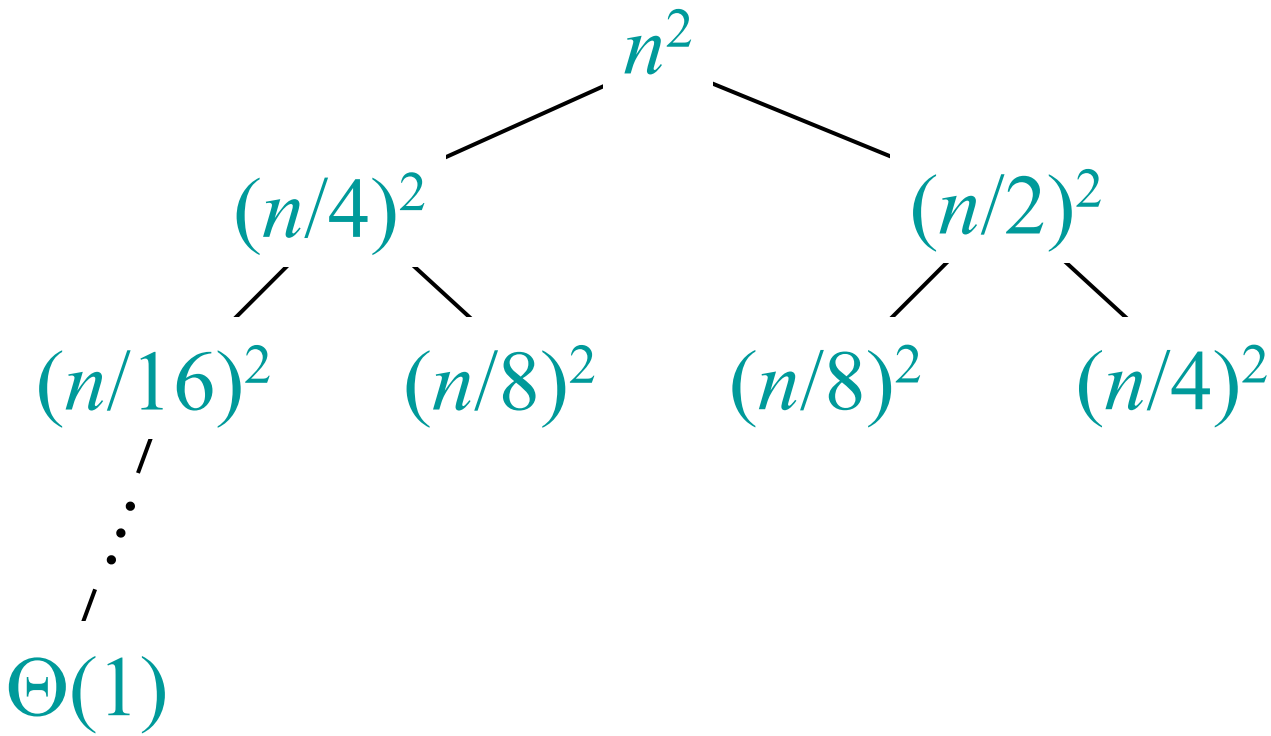
Solve $T(n) = T(n/4) + T(n/2) + n^2$:



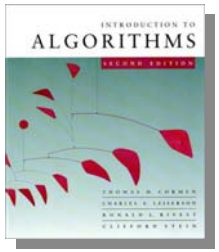



# Example of recursion tree

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

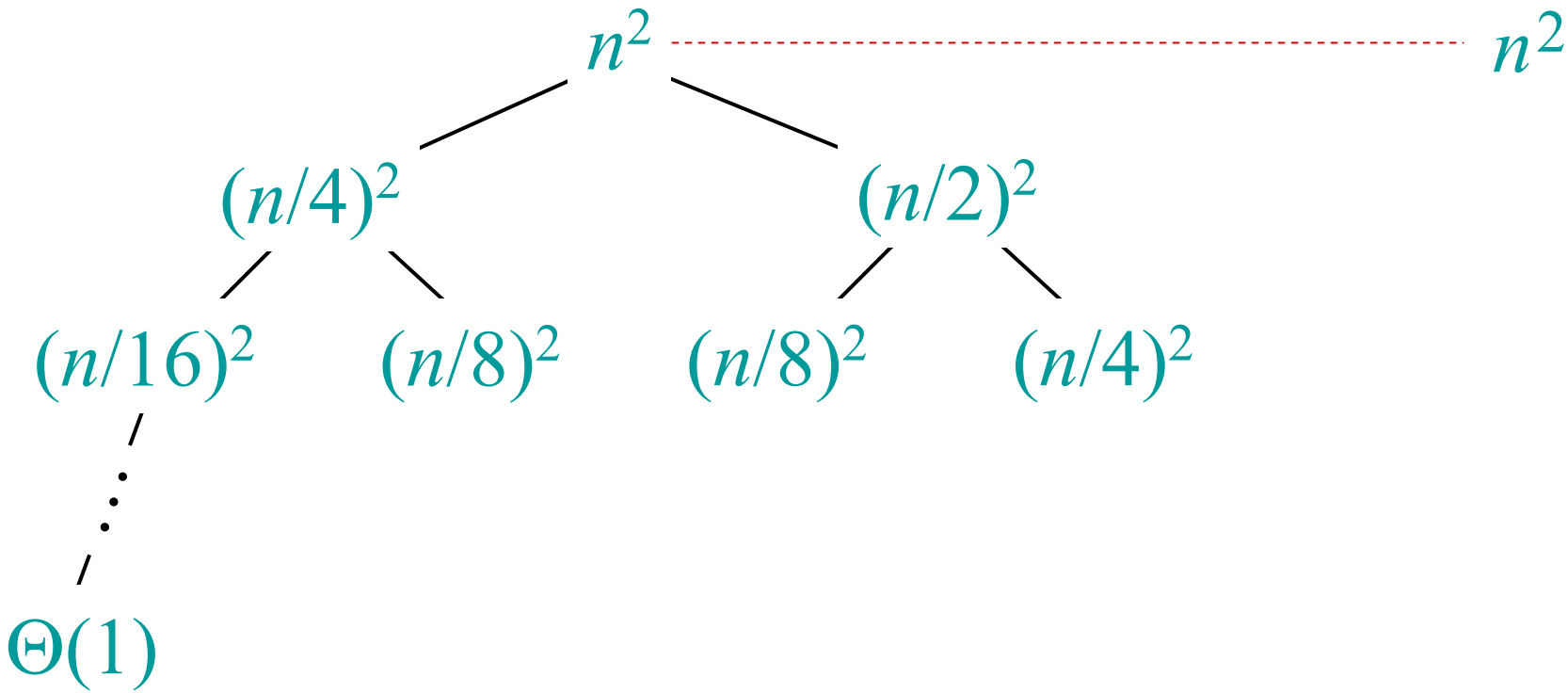


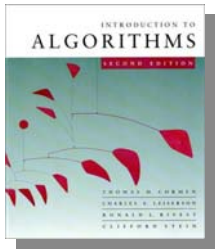



# Example of recursion tree

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

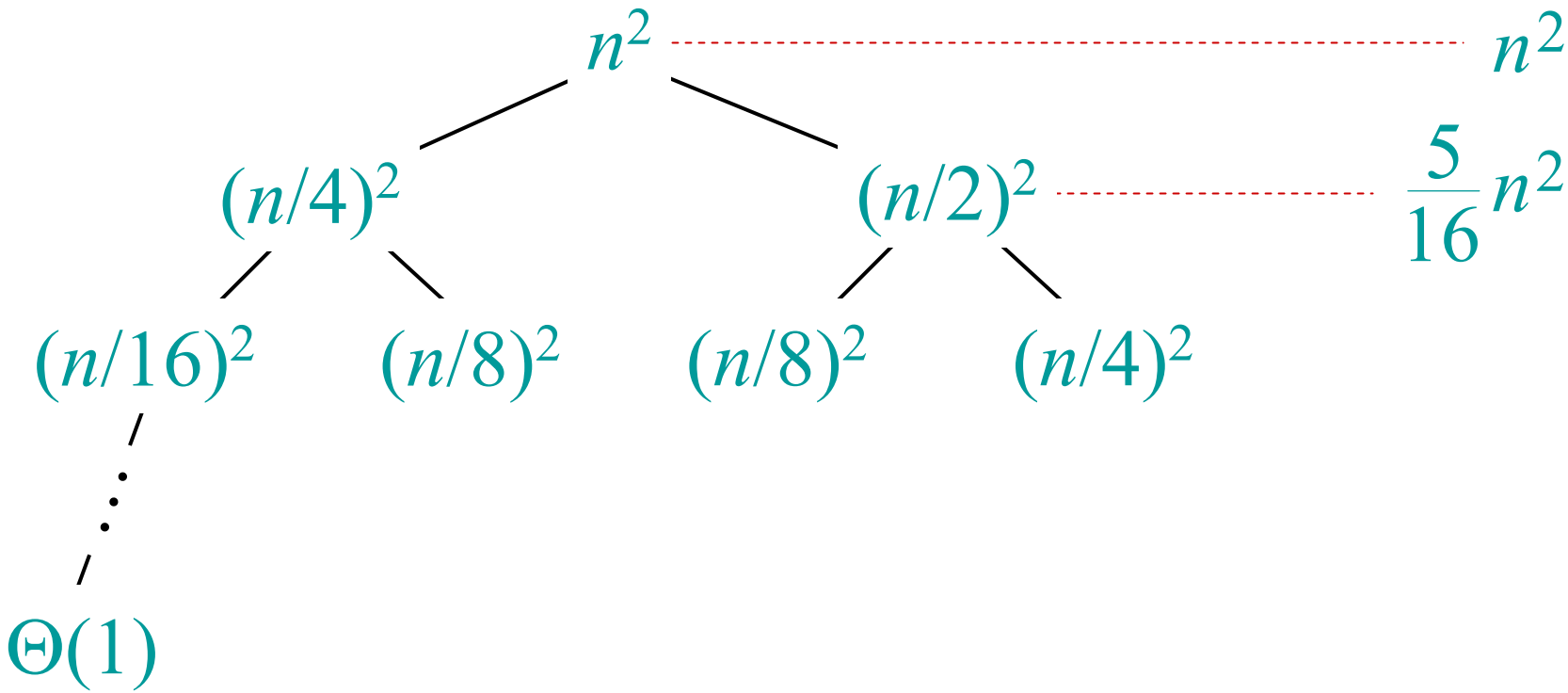


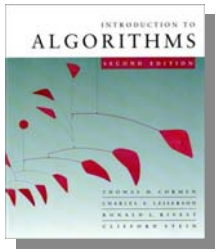



# Example of recursion tree

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

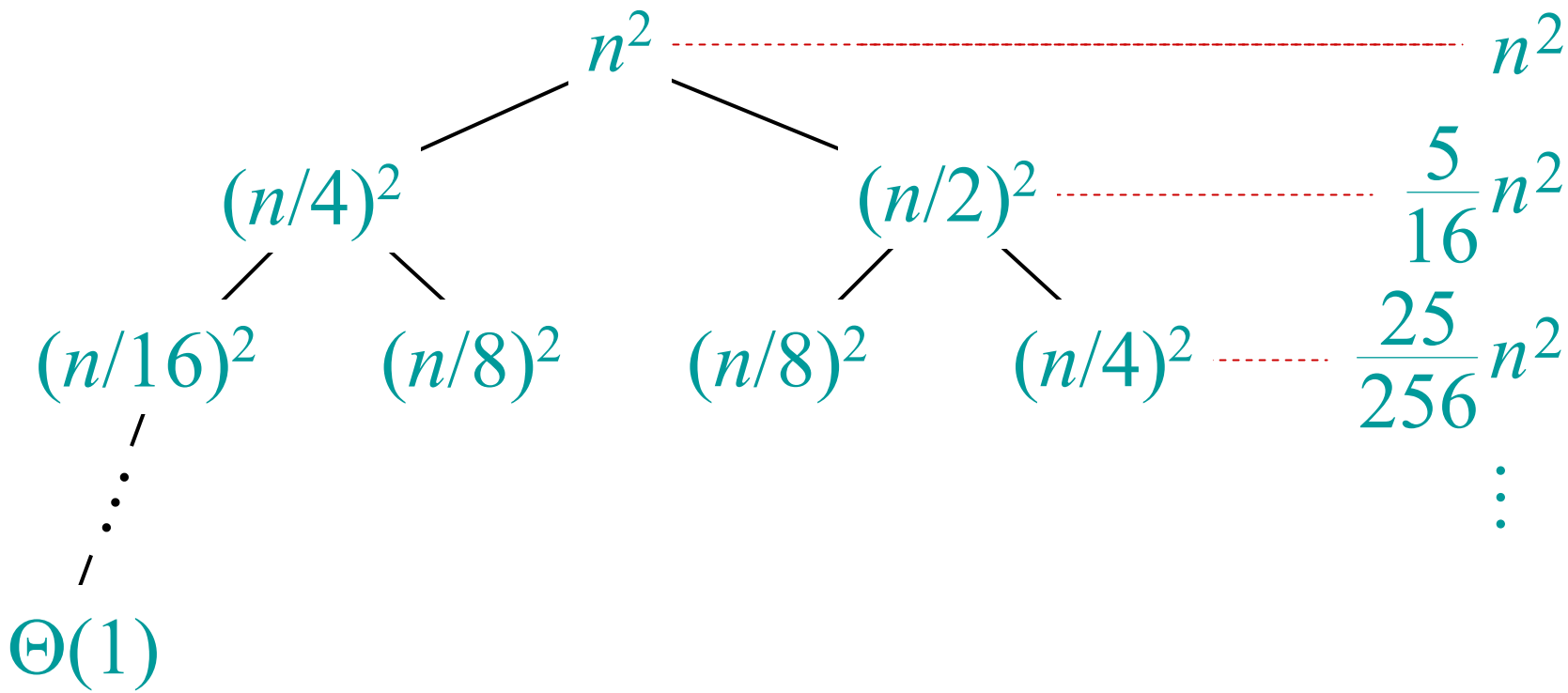


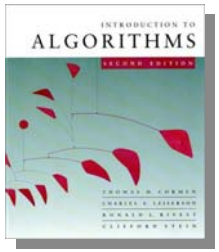



# Example of recursion tree

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

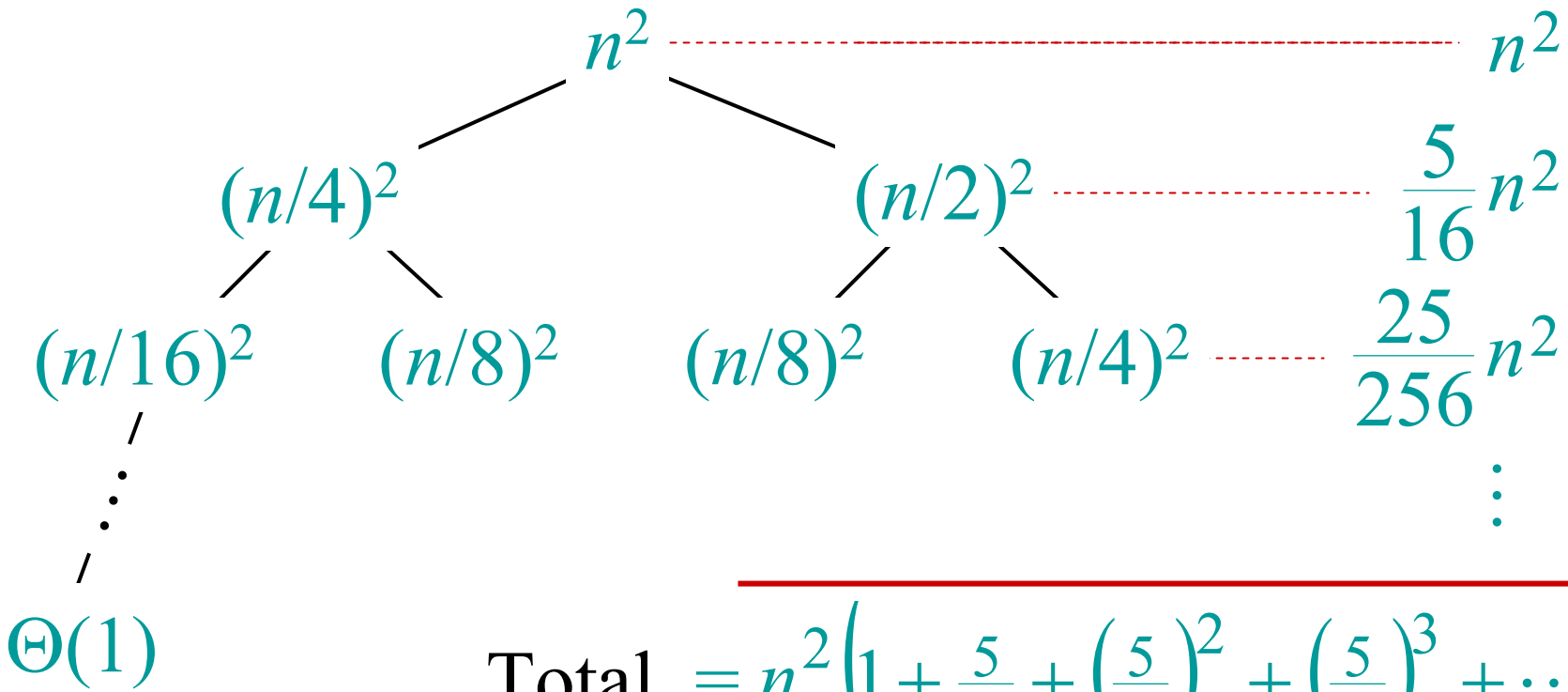





# Example of recursion tree

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




$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2) \quad \textit{geometric series}$$

### Arithmetic series

The summation

$$\sum_{k=1}^{n} k = 1 + 2 + \cdots + n \ ,$$

is an ***arithmetic series*** and has the value

$$\sum_{k=1}^{n} k = \frac{1}{2} n(n+1) \tag{A.1}$$

$$= \Theta(n^2) \ . \tag{A.2}$$

### Geometric series

For real $x \neq 1$, the summation

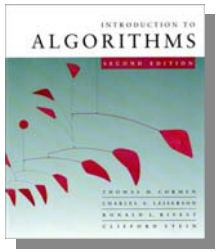
$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

is a ***geometric*** or ***exponential series*** and has the value

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1} \ . \tag{A.5}$$

When the summation is infinite and $|x| < 1$, we have the infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x} \ . \tag{A.6}$$




# Substitution method

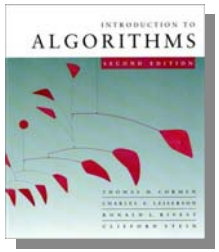
*The most general method:*

1. ***Guess*** the form of the solution.
2. ***Verify*** by induction.
3. ***Solve*** for constants.

**EXAMPLE:** $T(n) = 4T(n/2) + n$

- [Assume that $T(1) = \Theta(1)$.]
- Guess $O(n^3)$ . (Prove $O$ and $\Omega$ separately.)
- Assume that $T(k) \le ck^3$ for $k < n$ .
- Prove $T(n) \le cn^3$ by induction.



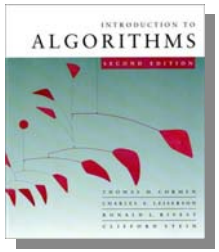

# Example of substitution

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

whenever $(c/2)n^3 - n \ge 0$, for example, if $c \ge 2$ and $n \ge 1$.

*residual*

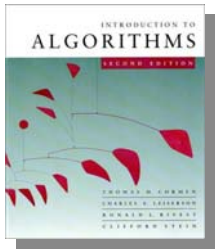
  




# Example (continued)

- We must also handle the initial conditions, that is, ground the induction with base cases.
- ***Base:*** $T(n) = \Theta(1)$ for all $n < n_0$, where $n_0$ is a suitable constant.
- For $1 \le n < n_0$, we have "$\Theta(1)$" $\le cn^3$, if we pick $c$ big enough.

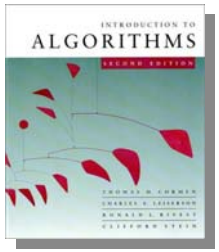



# Example (continued)

- We must also handle the initial conditions, that is, ground the induction with base cases.
- ***Base:*** $T(n) = \Theta(1)$ for all $n < n_0$, where $n_0$ is a suitable constant.
- For $1 \le n < n_0$, we have "$\Theta(1)$" $\le cn^3$, if we pick $c$ big enough.

***This bound is not tight!***

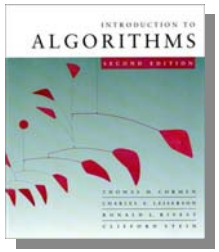
  




# A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.



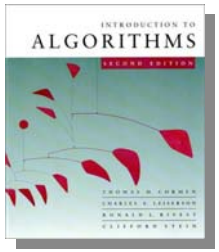

# A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \le ck^2$ for $k < n$:

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\le 4c(n/2)^2 + n \\ &= cn^2 + n \\ &= O(n^2) \end{aligned}$$




# A tighter upper bound?

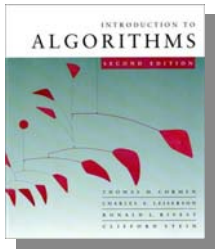
We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \le ck^2$ for $k < n$:

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\le 4c(n/2)^2 + n \\ &= cn^2 + n \\ &= O(n^2) \end{aligned}$$

***Wrong!*** We must prove the I.H.



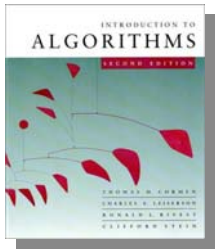

# A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \le ck^2$ for $k < n$:

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^2 + n \\
&= cn^2 + n \\
&= O(n^2) \quad \textbf{\textit{Wrong!}}\ \text{We must prove the I.H.} \\
&= cn^2 - (-n) \quad [\ \text{desired} - \text{residual}\ ] \\
&\le cn^2 \quad \text{for } \textbf{\textit{no}} \text{ choice of } c > 0. \ \text{Lose!}
\end{aligned}
$$



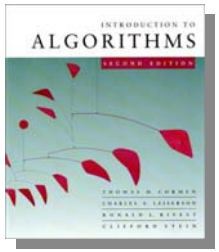

# A tighter upper bound!

**IDEA:** Strengthen the inductive hypothesis.

- ***Subtract*** a low-order term.

*Inductive hypothesis*: $T(k) \le c_1 k^2 - c_2 k$ for $k < n$.




# A tighter upper bound!

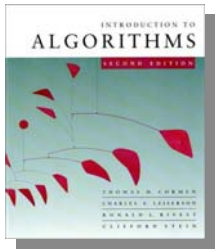
**IDEA:** Strengthen the inductive hypothesis.

- ***Subtract*** a low-order term.

*Inductive hypothesis*: $T(k) \le c_1 k^2 - c_2 k$ for $k < n$.

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




# A tighter upper bound!

**IDEA:** Strengthen the inductive hypothesis.

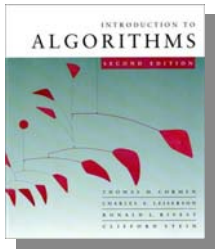
- ***Subtract*** a low-order term.

*Inductive hypothesis*: $T(k) \le c_1 k^2 - c_2 k$ for $k < n$.

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

Pick $c_1$ big enough to handle the initial conditions.



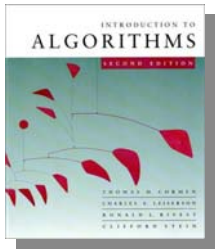

# The master method

The master method applies to recurrences of the form

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

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



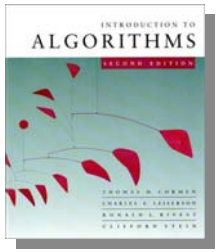

# Three common cases

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 $n^{\varepsilon}$ factor).

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




# Three common cases

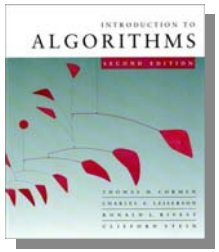
Compare $f(n)$ with $n^{\log_b a}$:

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 $n^{\varepsilon}$ 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.)

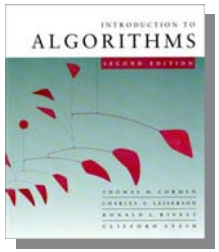
Compare $f(n)$ with $n^{\log_b a}$:

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 $n^{\varepsilon}$ factor),

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

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




# Examples

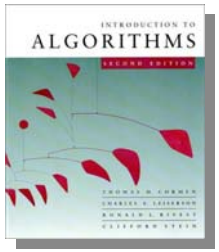
**Ex.** $T(n) = 4T(n/2) + n$

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

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

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




# Examples

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

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

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

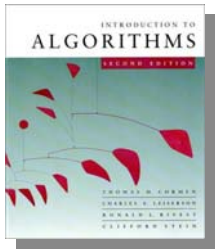
$\therefore\ T(n) = \Theta(n^2).$

**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

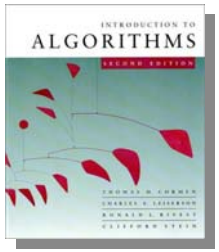
**Ex.** $T(n) = 4T(n/2) + n^3$

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

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

***and*** $4(n/2)^3 \le cn^3$ (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**: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$

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

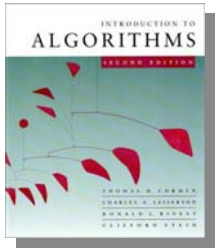
$\therefore\ T(n) = \Theta(n^3)$.

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

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

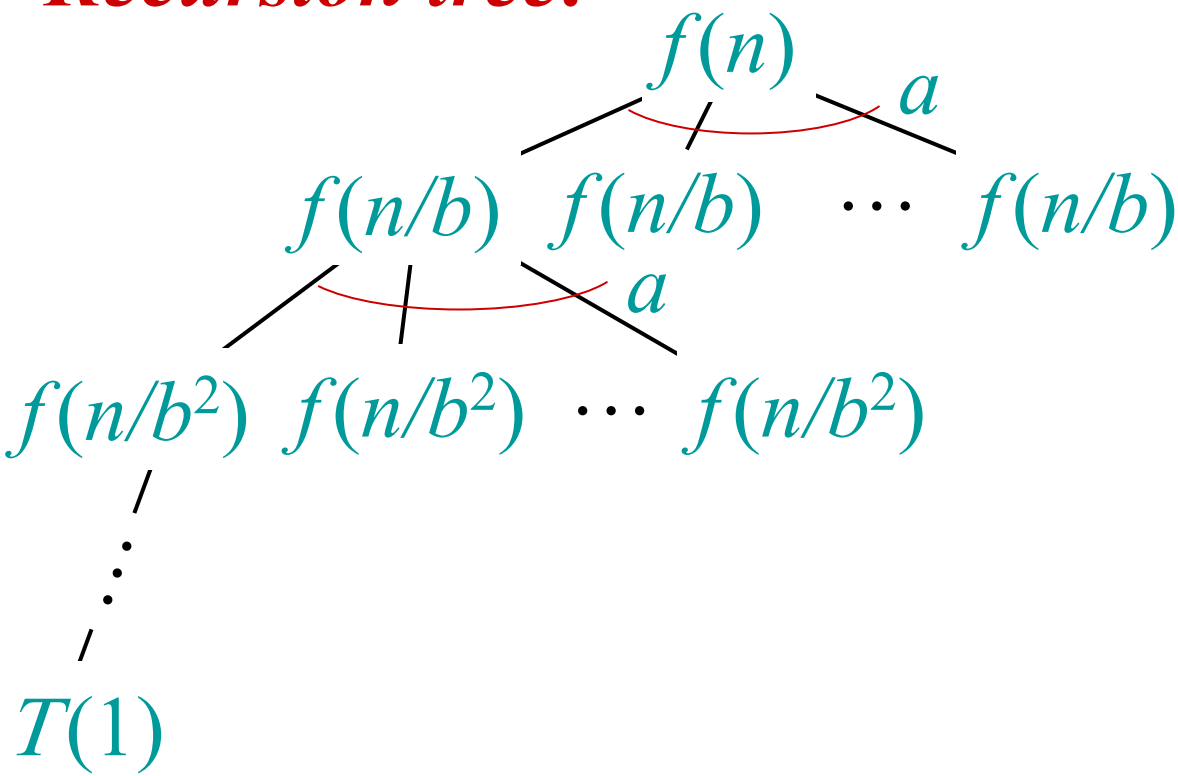
Master method does not apply. In particular, for every constant $\varepsilon > 0$, we have $n^\varepsilon = \omega(\lg n)$.

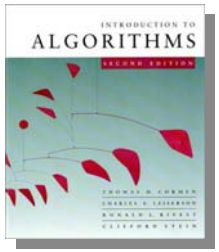



# Idea of master theorem

***Recursion tree:***

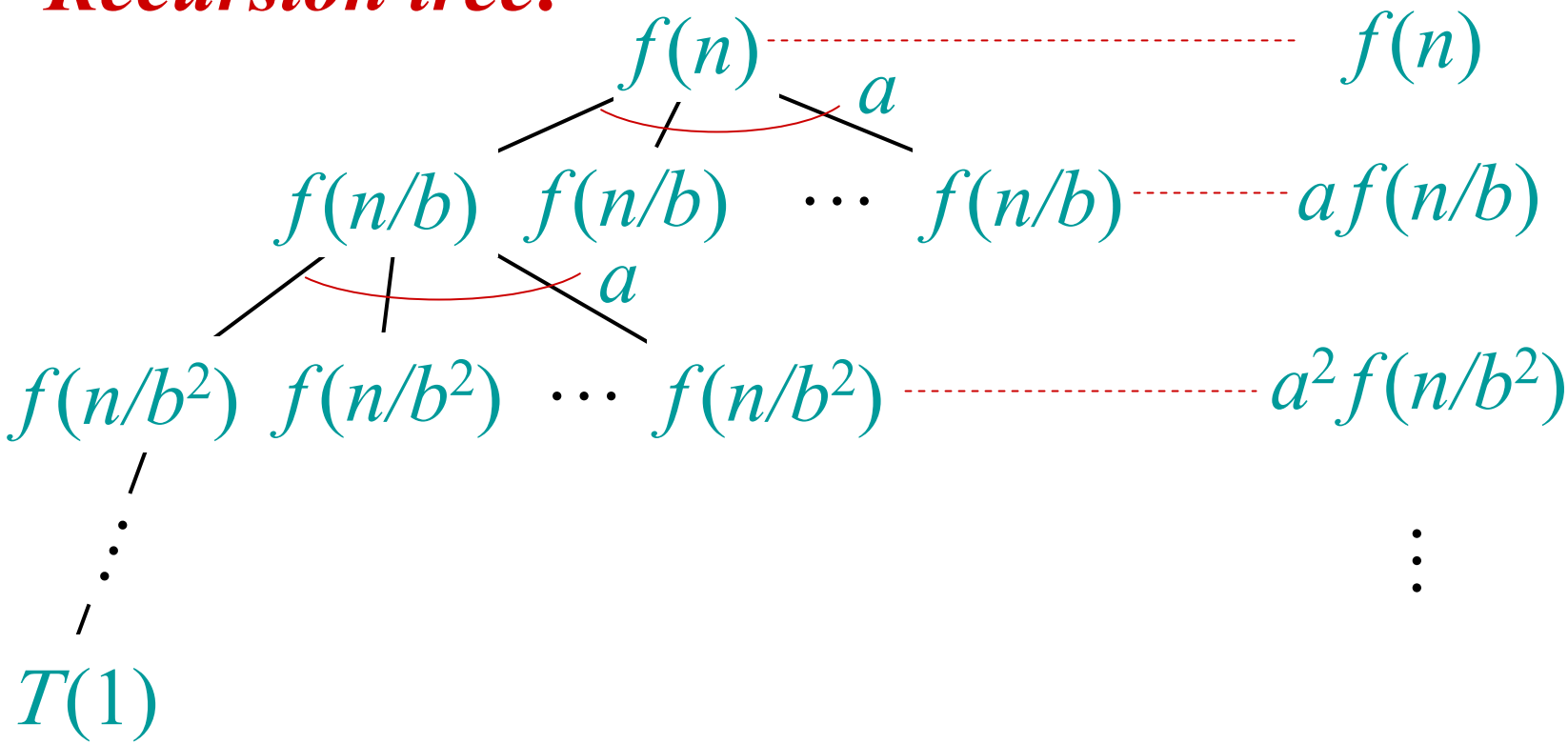


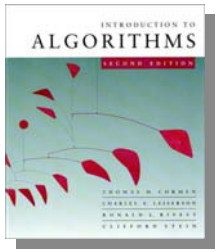



# Idea of master theorem

***Recursion tree:***

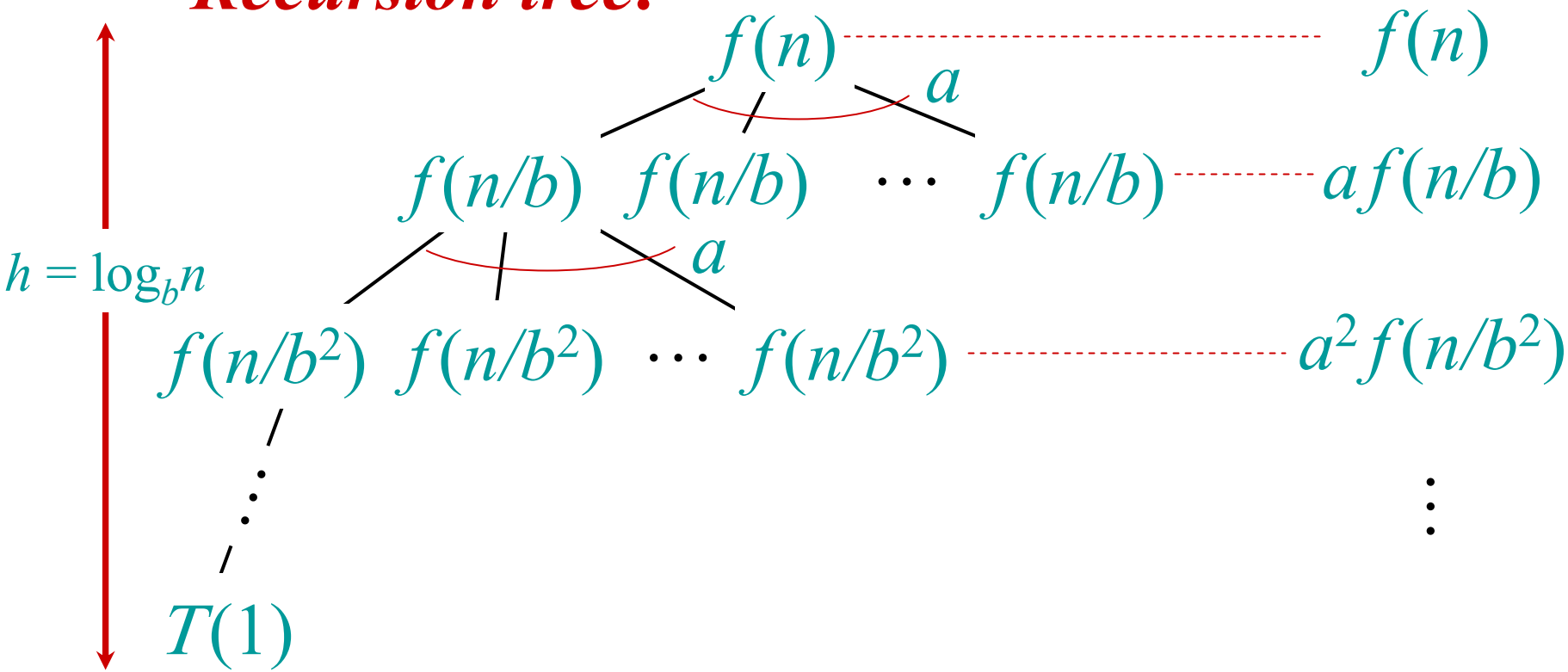


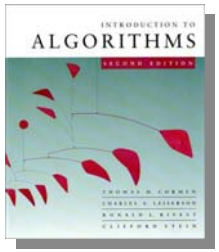



# Idea of master theorem

***Recursion tree:***

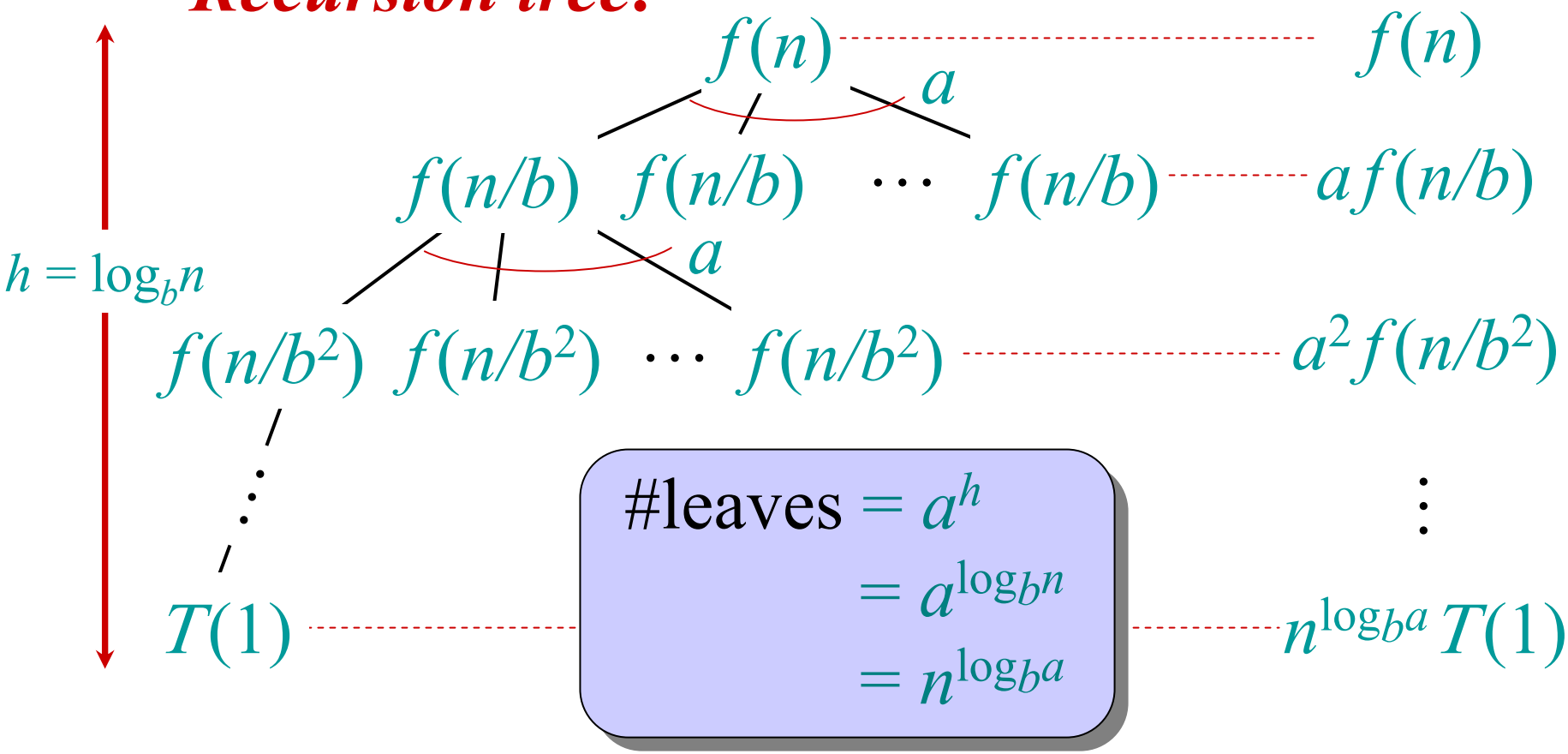


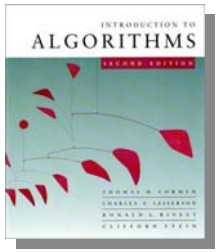



# Idea of master theorem

***Recursion tree:***

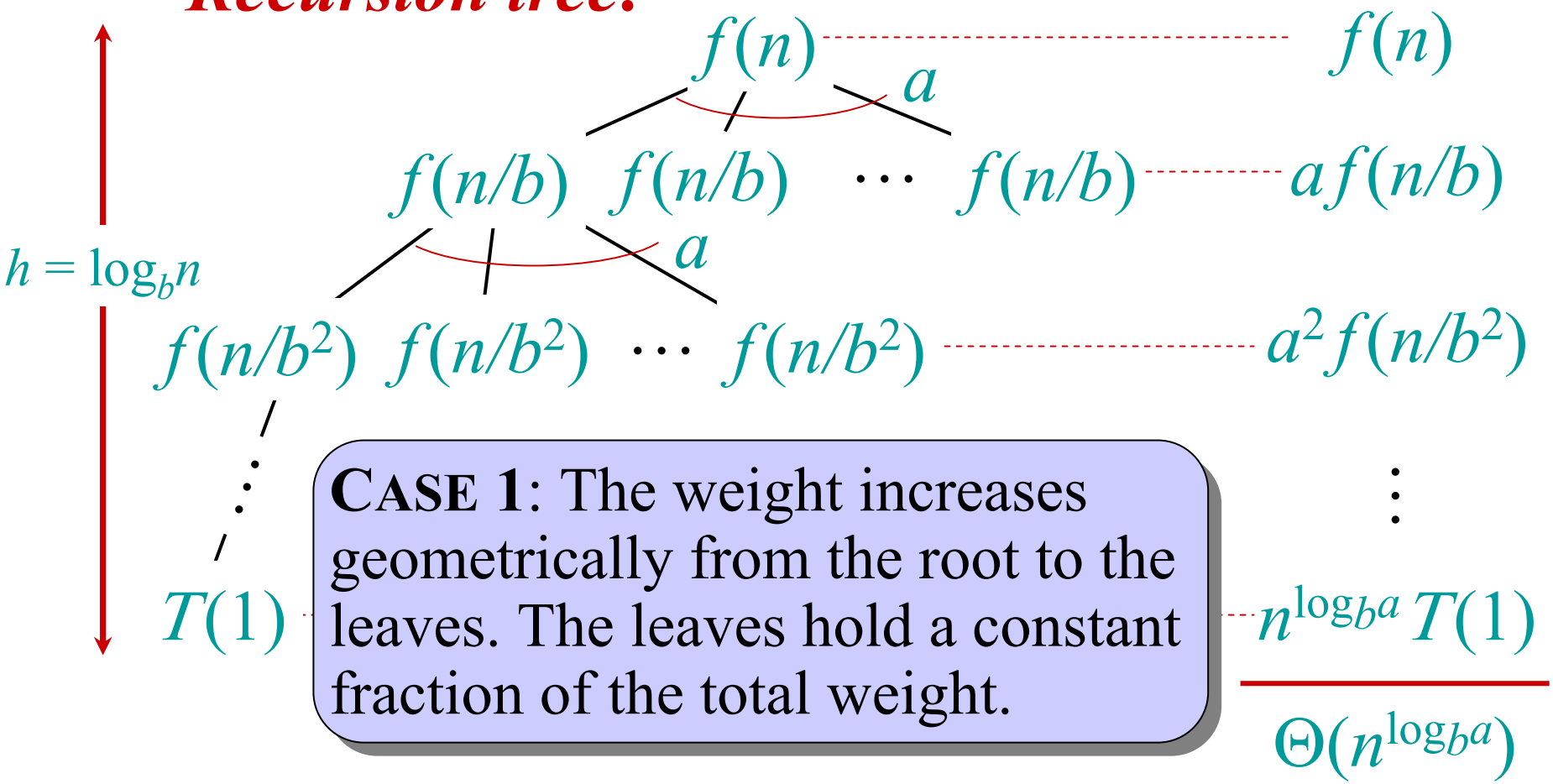




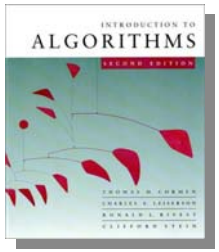

# Idea of master theorem

***Recursion tree:***



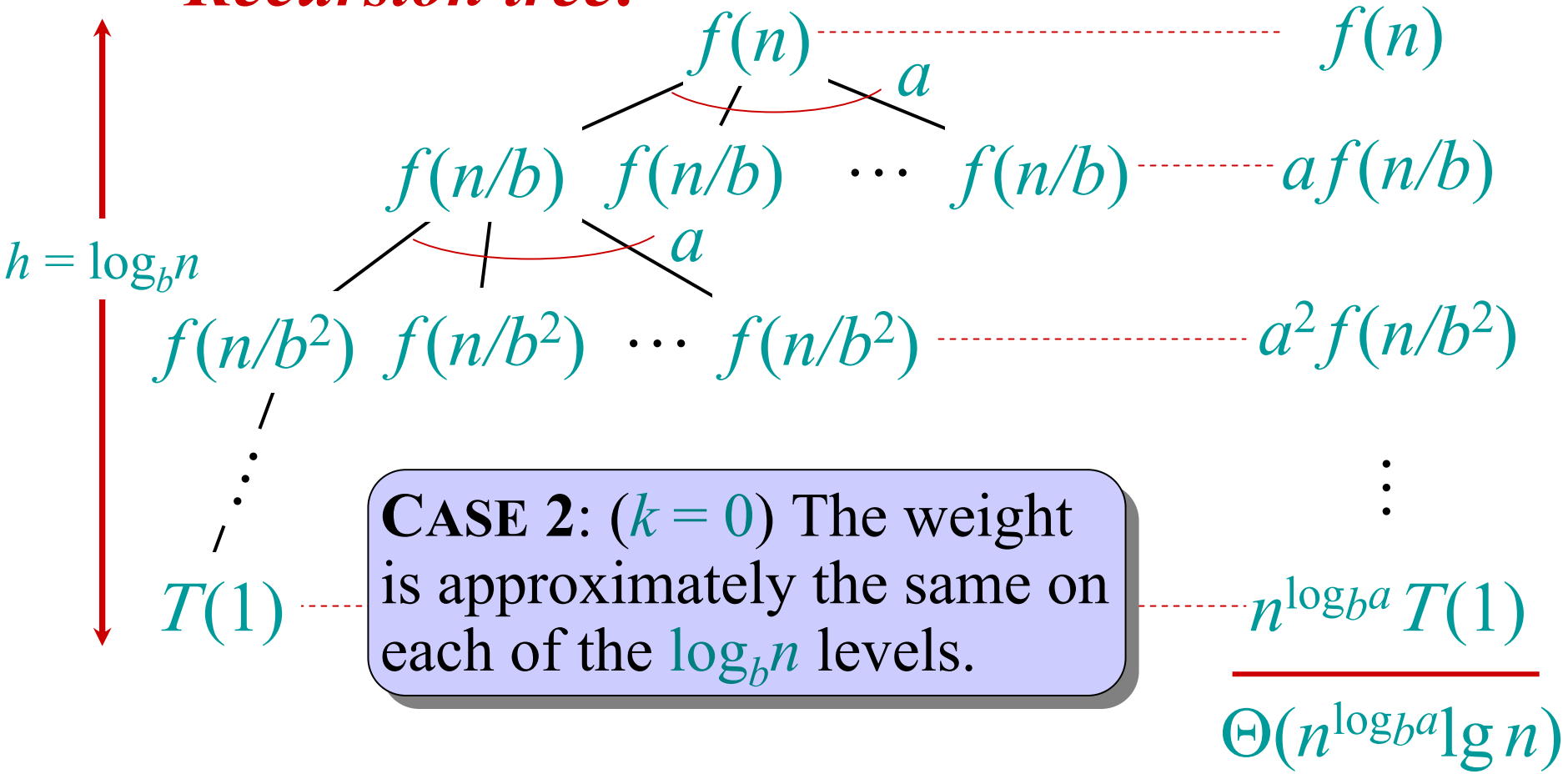

**CASE 1**: The weight increases geometrically from the root to the leaves. The leaves hold a constant fraction of the total weight.



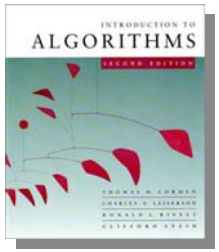

# Idea of master theorem

***Recursion tree:***



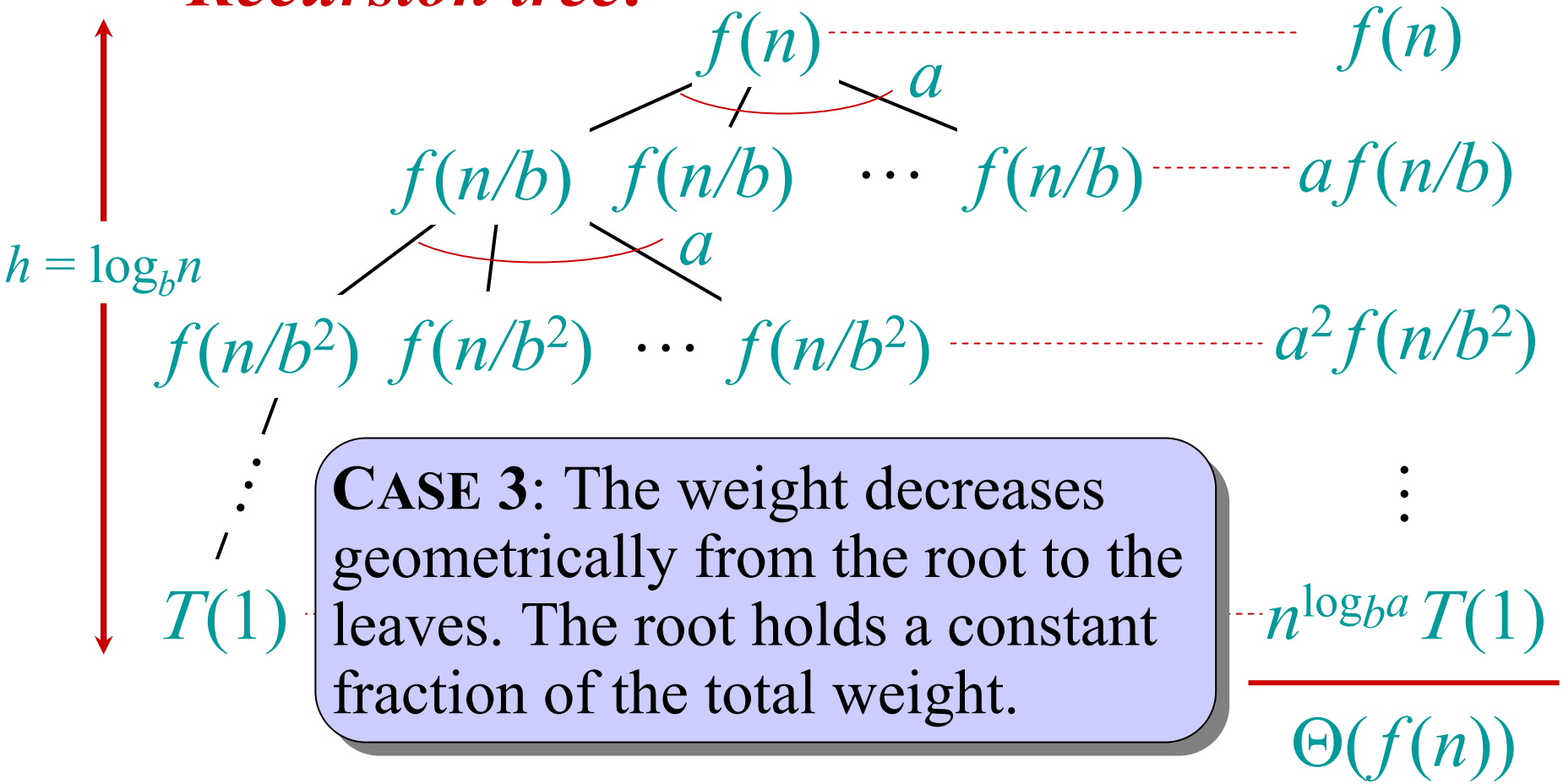

**CASE 2**: ($k = 0$) The weight is approximately the same on each of the $\log_b n$ levels.



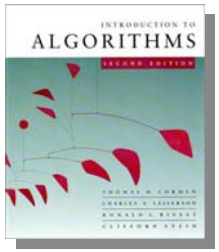

# Idea of master theorem

***Recursion tree:***




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




# Appendix: geometric series

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

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

Return to last slide viewed.