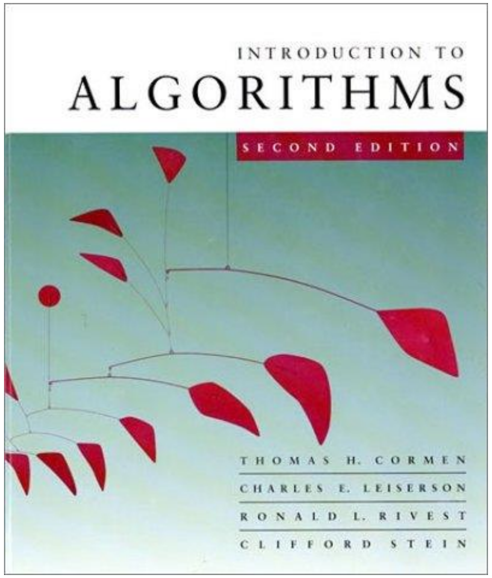
# *Divide-and-Conquer Algorithm: An Idea of Master Theorem*



# Integer Multiplication

- Let X = $\boxed{A\,|\,B}$ and Y = $\boxed{C\,|\,D}$ be the *n* bit integers, where A, B, C and D are *n*/2 bit integers
- Simple Method: $XY = (A.2^{n/2}+B)(C.2^{n/2}+D)$
- Running Time Recurrence Relation:

$$T(n) < 4T(n/2) + 100n$$

How do we solve it?

# Substitution Method

*The most general method:*

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

***Example:*** $T(n) = 4T(n/2) + 100n$

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

# Example of Substitution

$$
\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\leq 4c(n/2)^3 + 100n \\
&= (c/2)n^3 + 100n \\
&= cn^3 - ((c/2)n^3 - 100n) \longleftarrow \textit{Desired} - \textit{Residual} \\
&\leq cn^3 \longleftarrow \textit{Desired}
\end{aligned}
$$

whenever $(c/2)n^3 - 100n \geq 0$, for example, if $c \geq 200$ and $n \geq 1$.

*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.

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

# A Tighter Upper Bound?

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

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

$$
\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\le \quad cn^2 + 100n \\
&\le cn^2
\end{aligned}
$$

for ***no*** choice of $c > 0$. Lose!

# 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 $k < n$.

$$\begin{aligned} T(n) &= 4T(n/2) + 100n \\ &\le 4(c_1(n/2)^2 - c_2(n/2)) + 100n \\ &= c_1 n^2 - 2c_2 n + 100n \\ &= c_1 n^2 - c_2 n - (c_2 n - 100n) \\ &\le c_1 n^2 - c_2 n \quad \text{if } c_2 > 100. \end{aligned}$$

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

# Recursion-Tree Method

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

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

$n^2$

$T(n/4)$ $\qquad\qquad$ $T(n/2)$

# Example of Recursion Tree

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

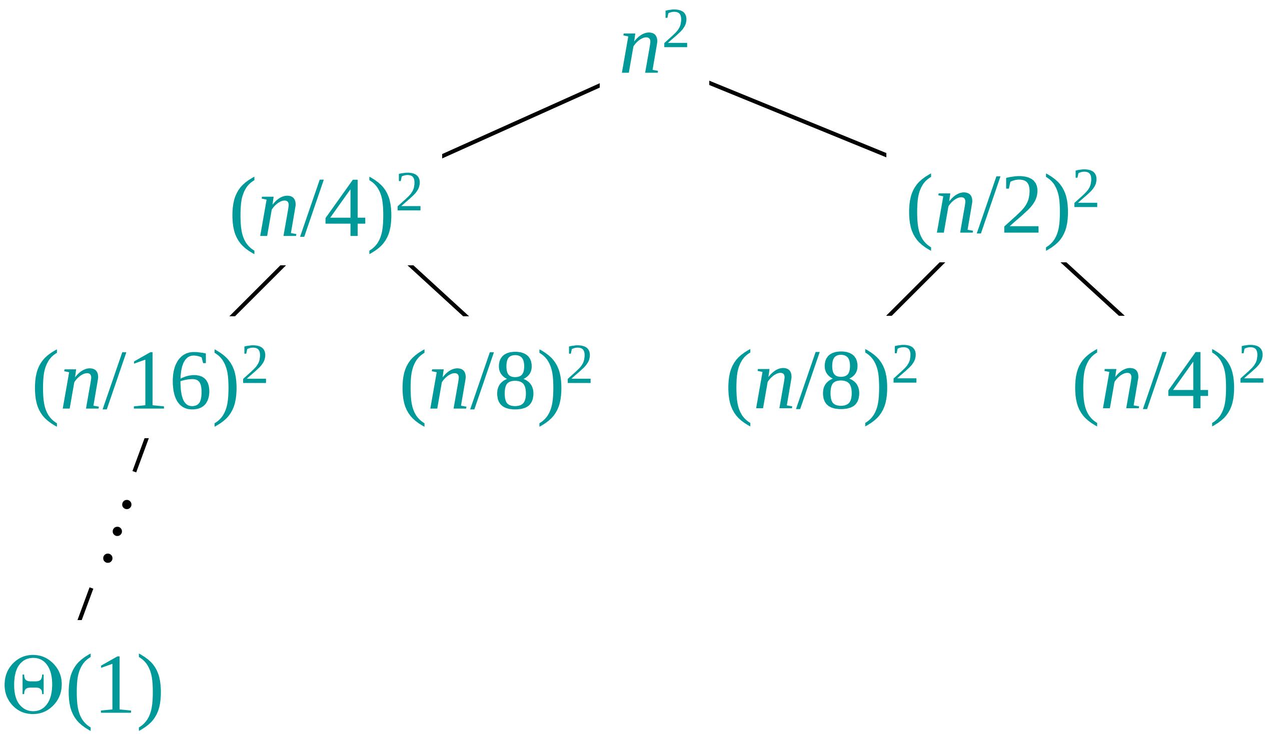
$$n^2$$

$$(n/4)^2 \qquad\qquad (n/2)^2$$

$$T(n/16) \qquad T(n/8) \qquad T(n/8) \qquad T(n/4)$$

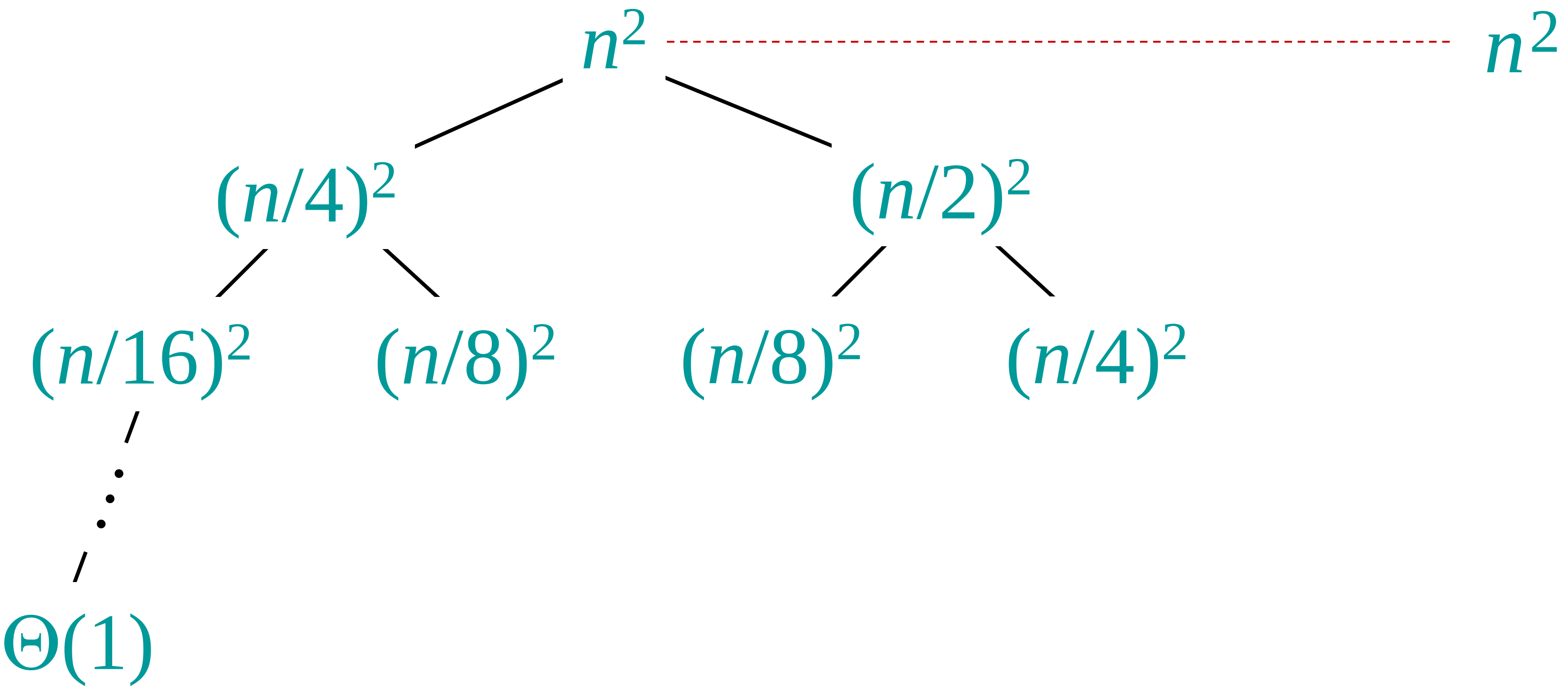
# Example of Recursion Tree

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



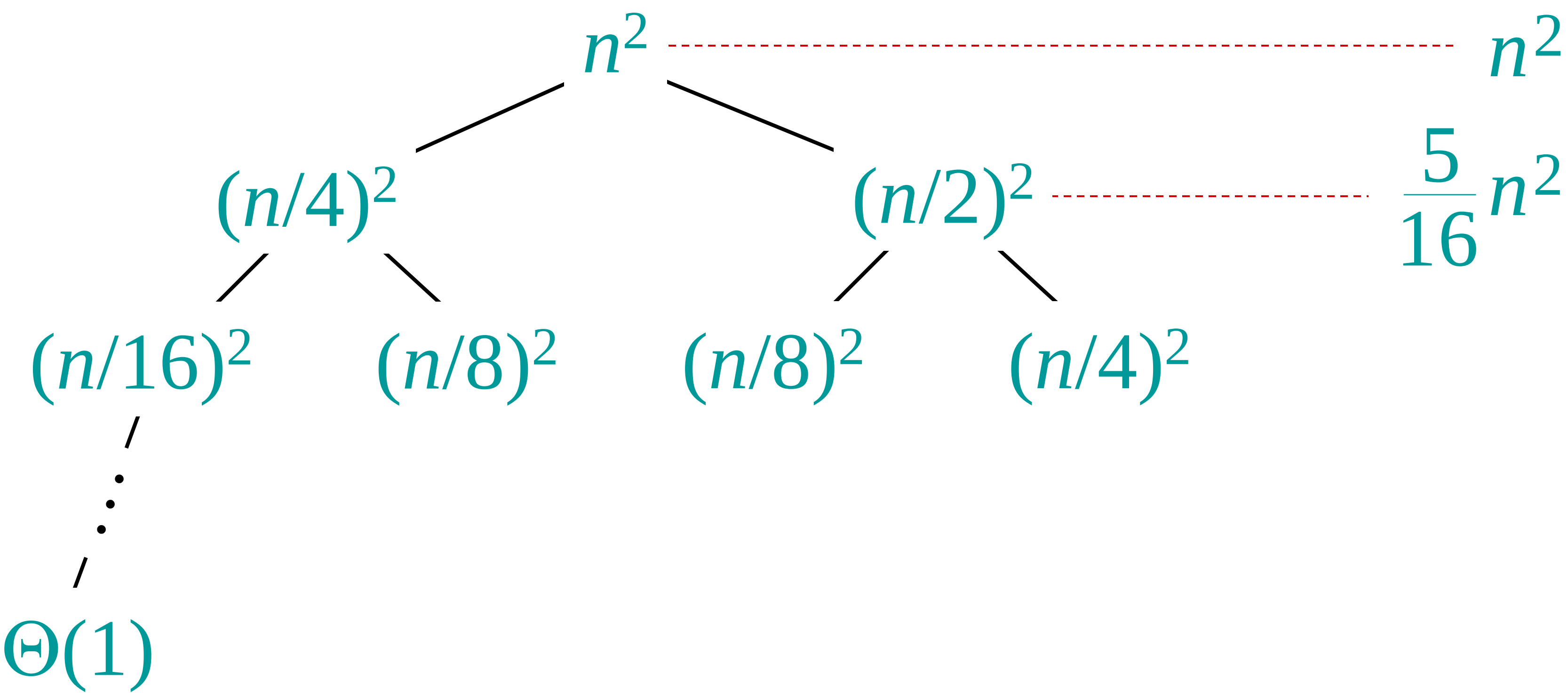
# Example of Recursion Tree

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



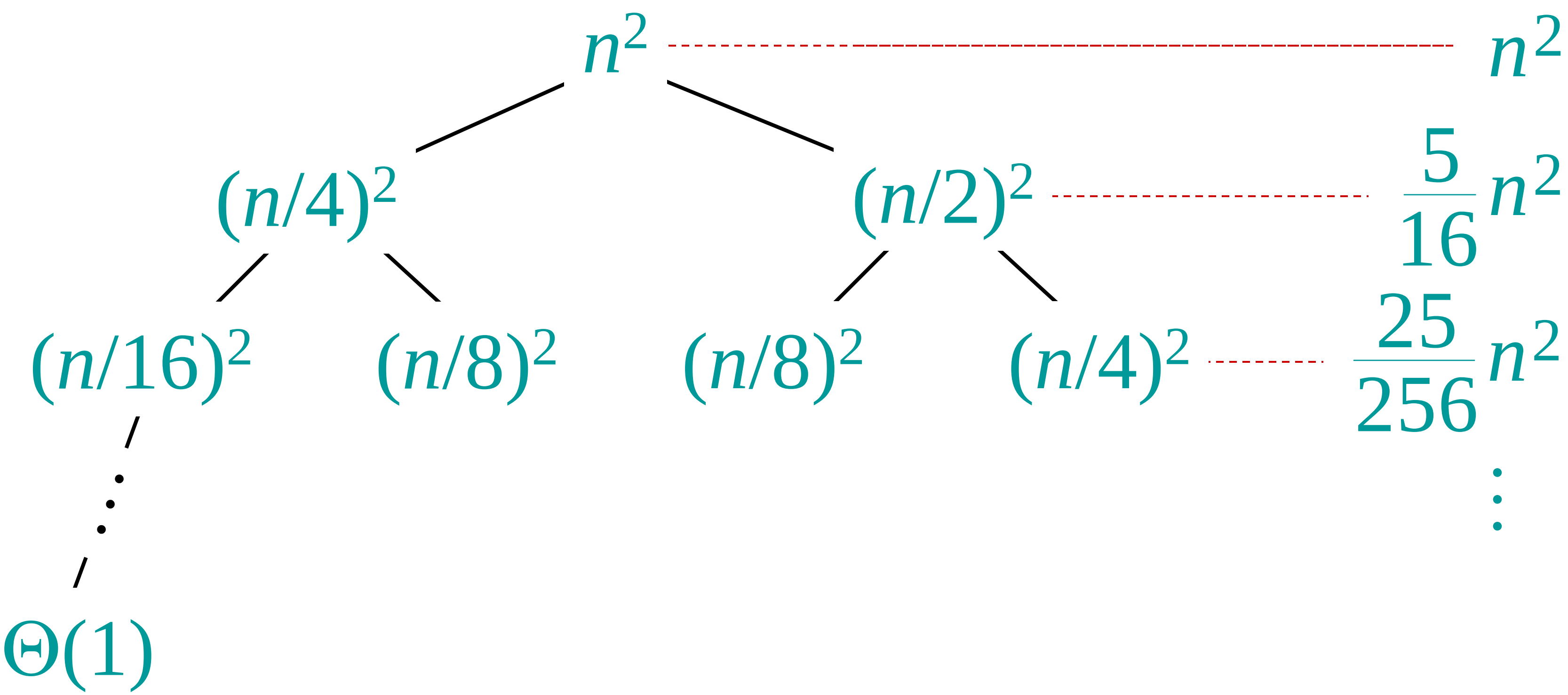
# Example of Recursion Tree

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



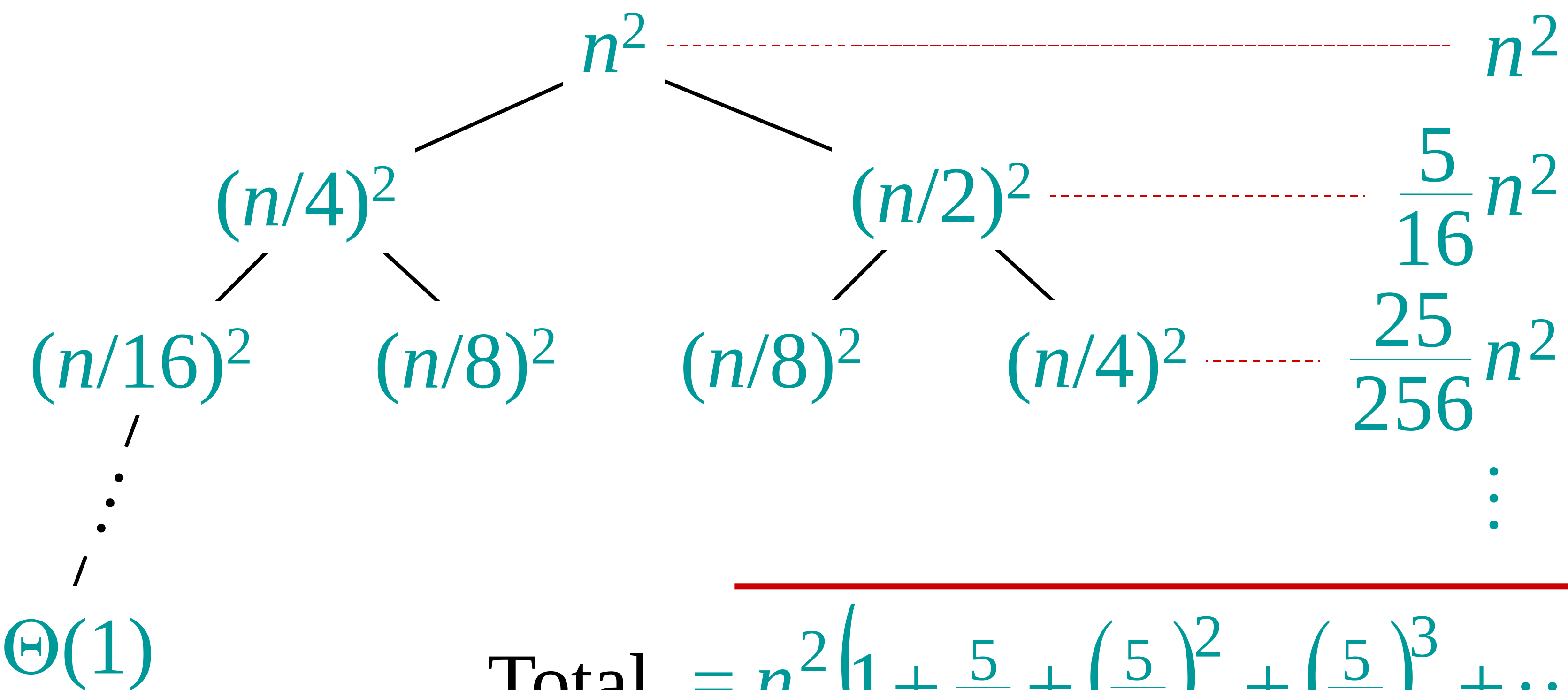
# Example of Recursion Tree

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



# Example of Recursion Tree

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




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

$$= \Theta(n^2)$$

*Geometric Series*

# Appendix: Geometric Series

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

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

# The Master Method

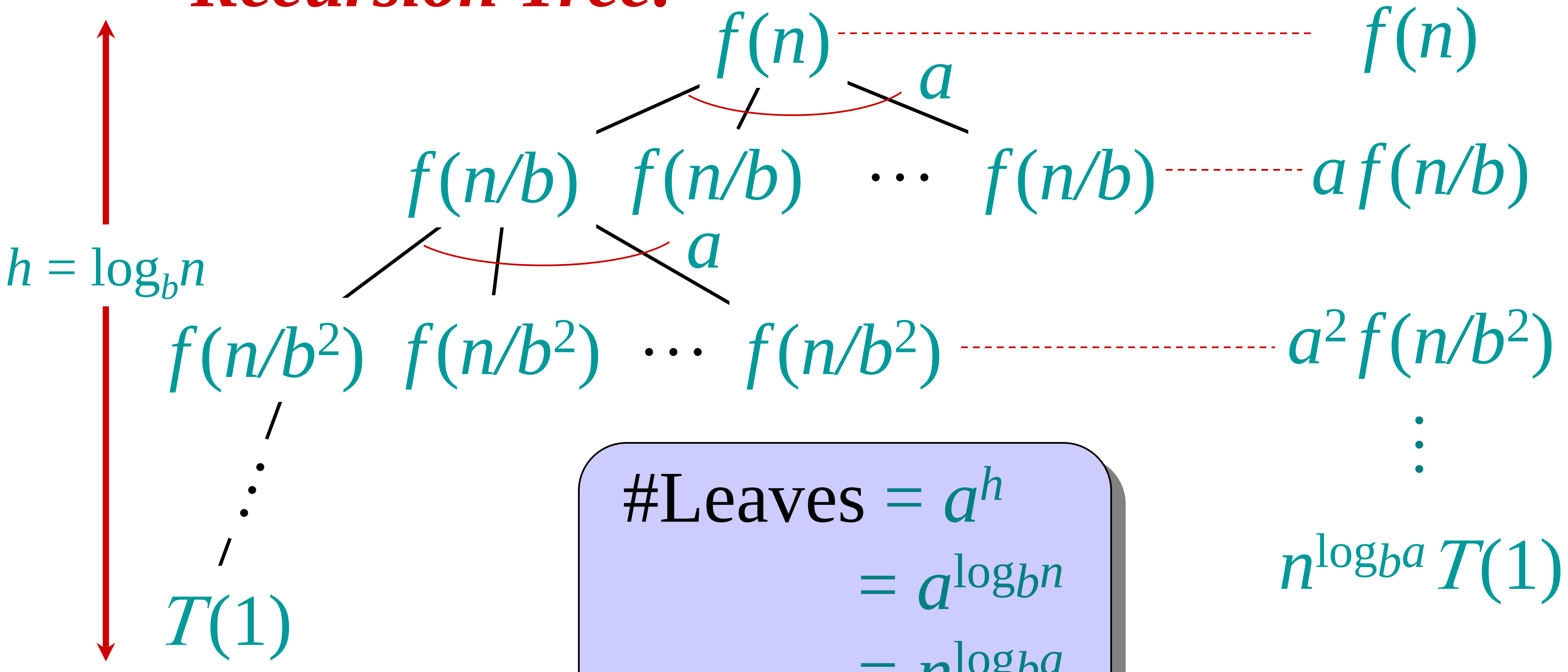
The master method applies to recurrences of the form

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

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

# Idea of Master Theorem

***Recursion Tree:***




$$\#\text{Leaves} = a^h = a^{\log_b n} = n^{\log_b a}$$

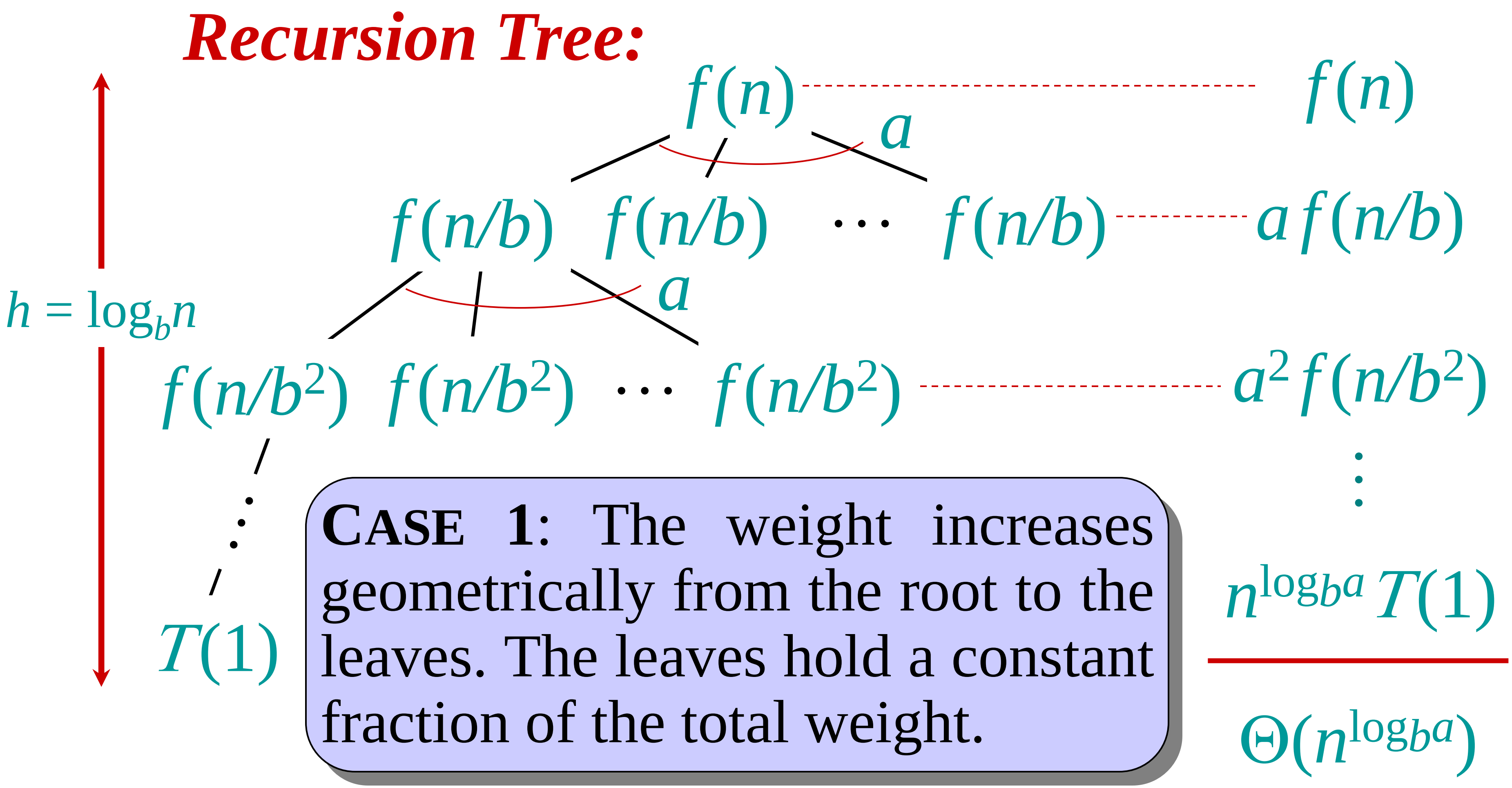
# Three Common Cases

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

# Idea of Master Theorem




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

# Three Common Cases

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

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

# Idea of Master Theorem

***Recursion Tree:***

$f(n)$ — $f(n)$

$a$

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

$a$

$h = \log_b n$

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

$\vdots$

$T(1)$ — $n^{\log_b a}\,T(1)$

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

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

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

# Idea of Master Theorem

***Recursion Tree:***

$$f(n) \quad\cdots\cdots\quad f(n)$$

$$a$$

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

$$a$$

$$h = \log_b n$$

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

$$\vdots$$

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

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

$$\Theta(f(n))$$

# 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 $\varepsilon = 1$.

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

***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*** $4(cn/2)^3 \le cn^3$ (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 $n^\varepsilon = \omega(\lg n)$.