Lecture 2: Solving Recurrences

SOLVING RECURRENCES

- The analysis of merge sort from ***Lecture 1*** required us to solve a recurrence.
- Merge Sort recurrence:

  $T(n) = 2T(n/2) + \Theta(n)$
- How to solve a recurrence?

SUBSTITUTION METHOD

The most general method:

1. ***Guess*** the form of the solution.
2. ***Verify*** by induction.

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

- Guess $O(n^3)$.
- Assume that $T(k) \leq ck^3$ for $k < n$.
- Prove $T(n) \leq cn^3$ by induction.

EXAMPLE OF SUBSTITUTION

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

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

A TIGHTER UPPER BOUND?

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

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

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

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 $k < n$.

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

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 promotes intuition, however.

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

$$
\begin{array}{ccc}
 & n^2 & \\
\swarrow & & \searrow \\
T(n/4) & & T(n/2)
\end{array}
$$

EXAMPLE OF RECURSION TREE

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

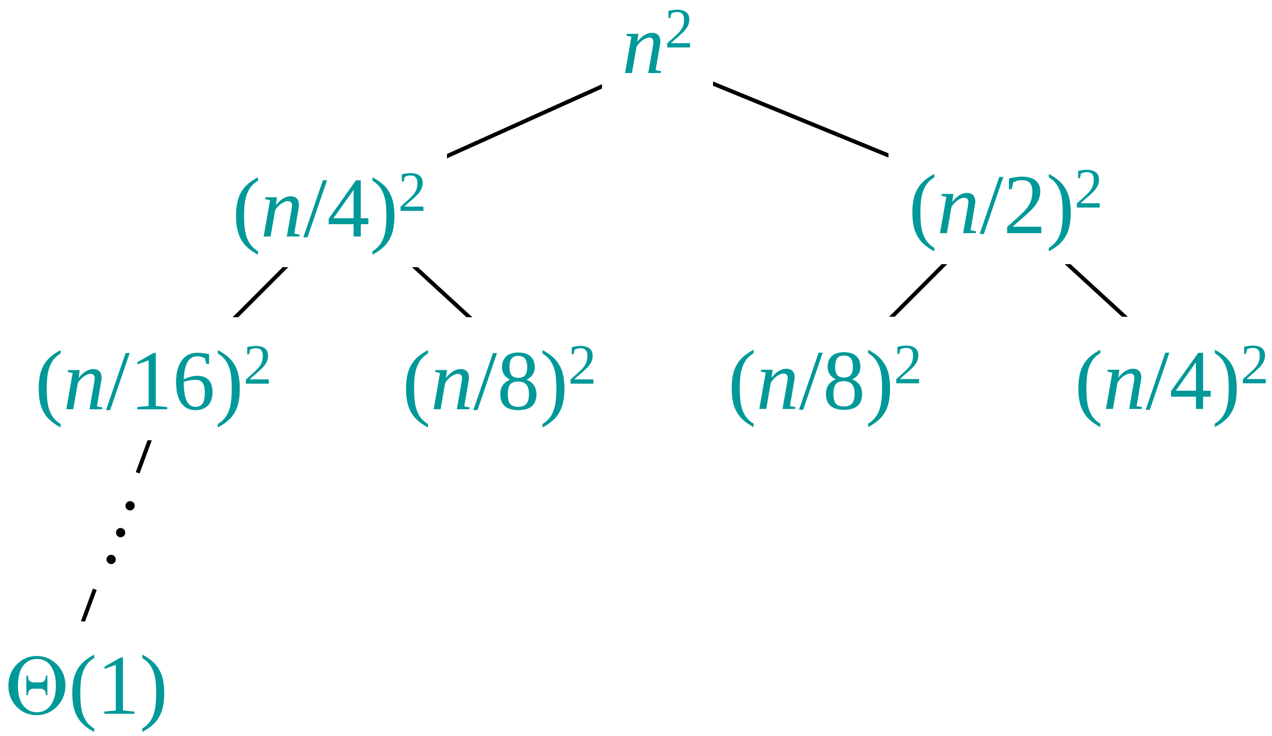
$$n^2$$

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

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

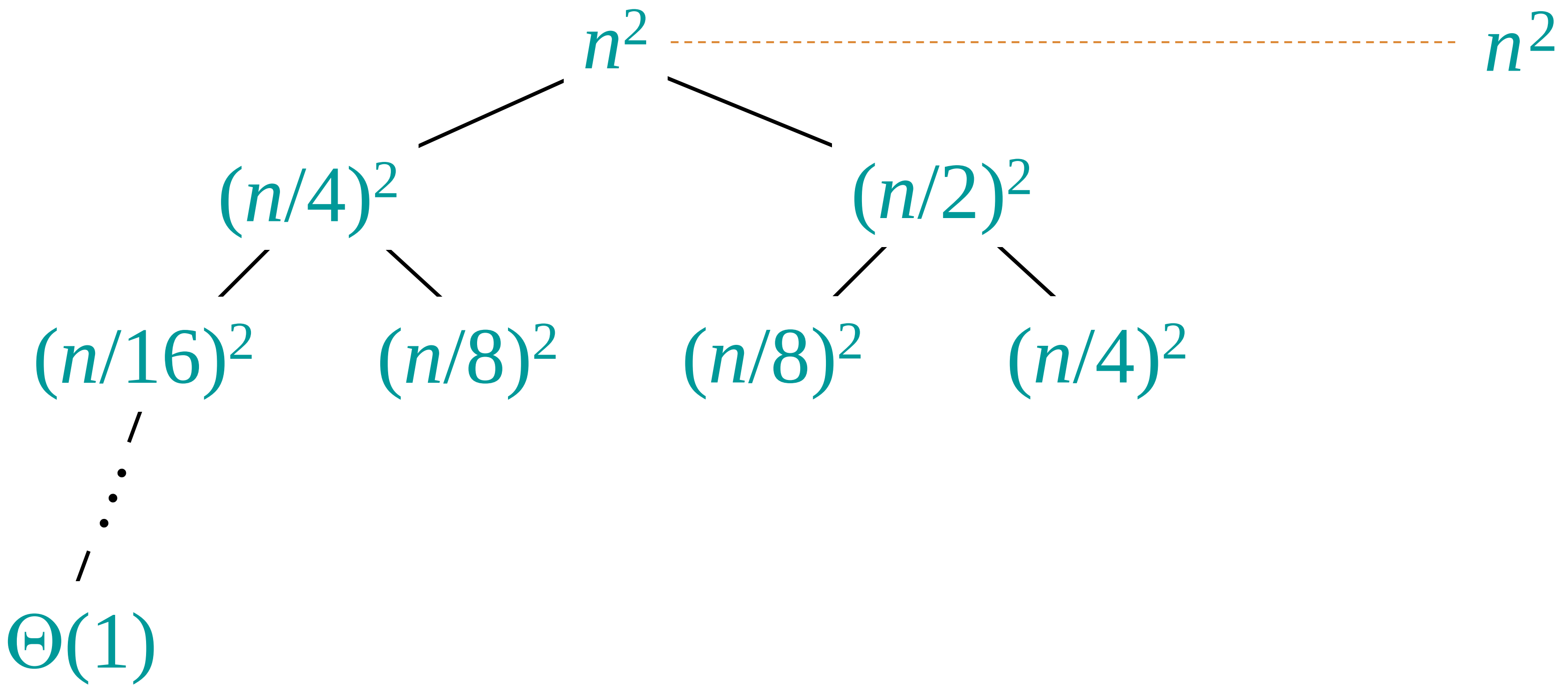
EXAMPLE OF RECURSION TREE

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



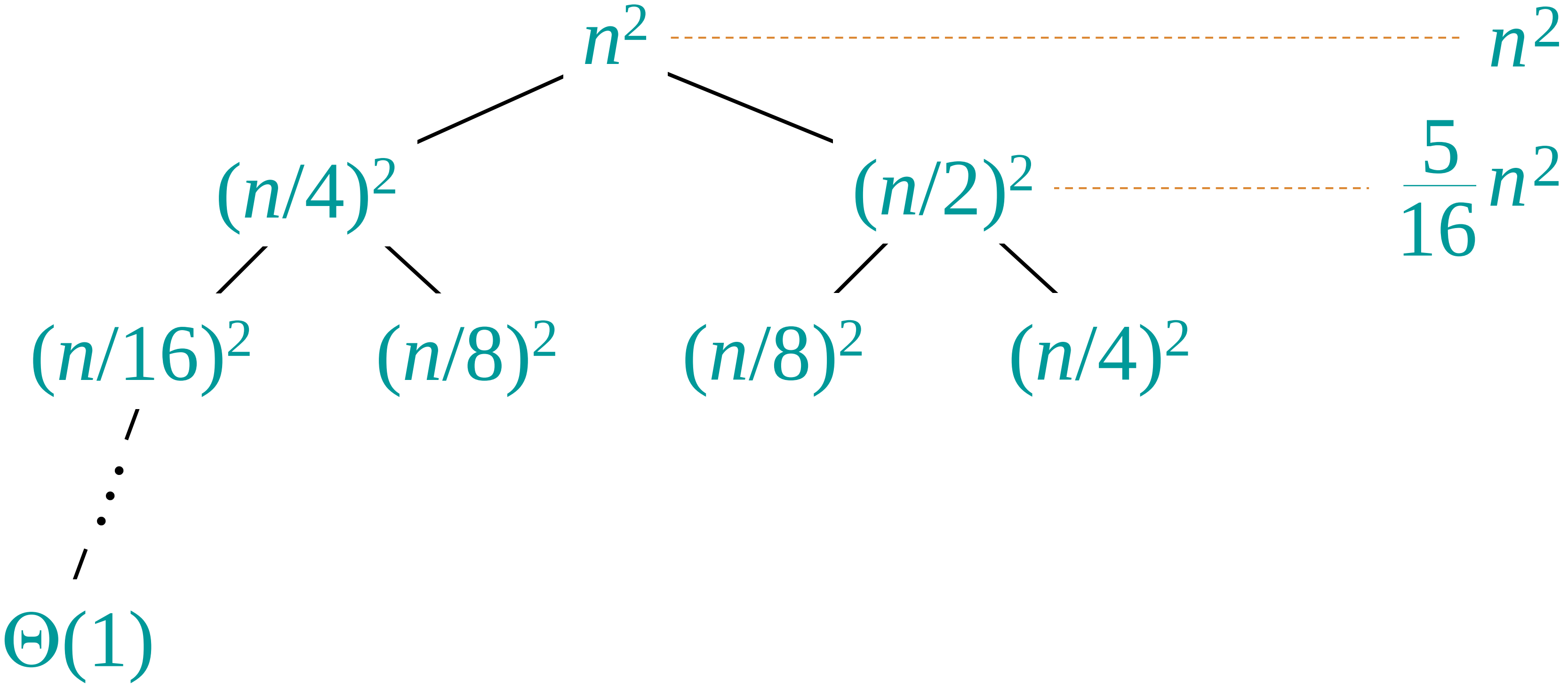
EXAMPLE OF RECURSION TREE

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



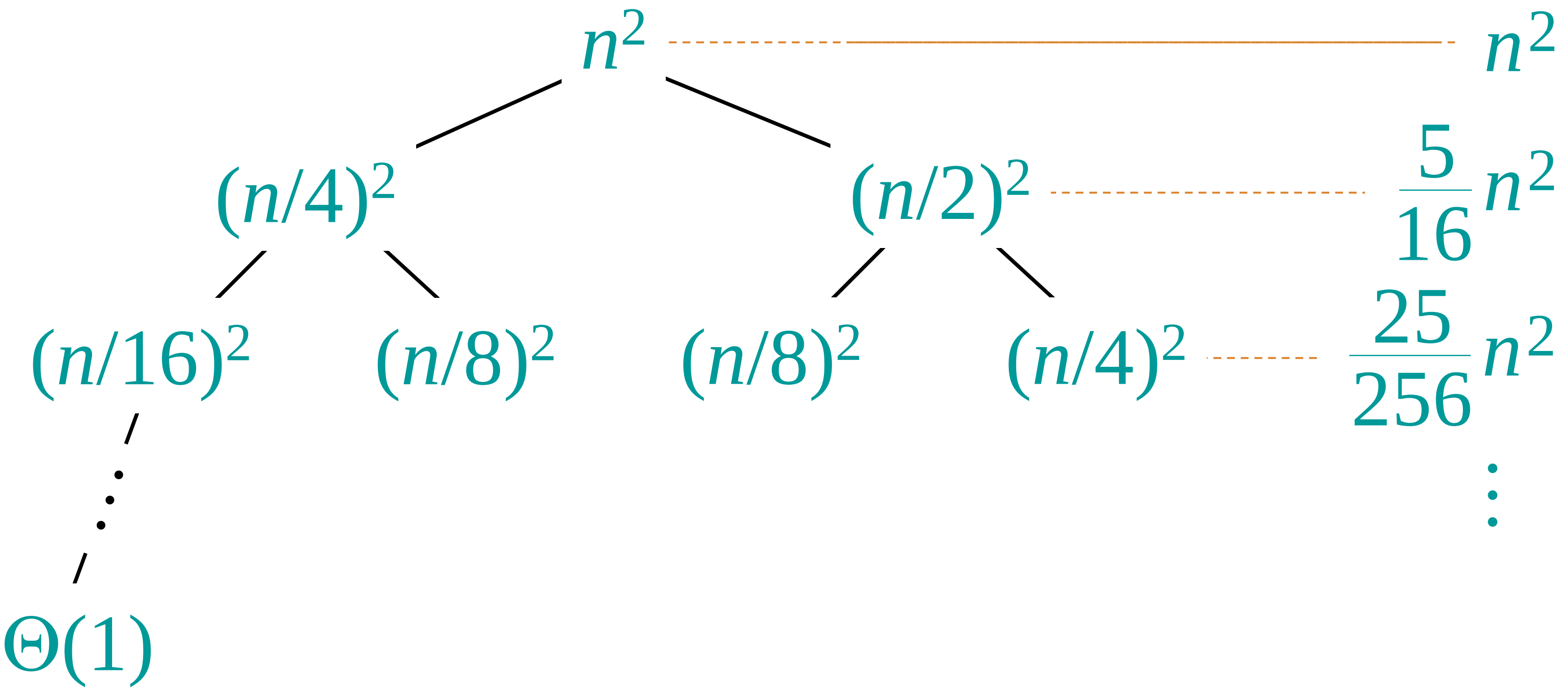
EXAMPLE OF RECURSION TREE

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



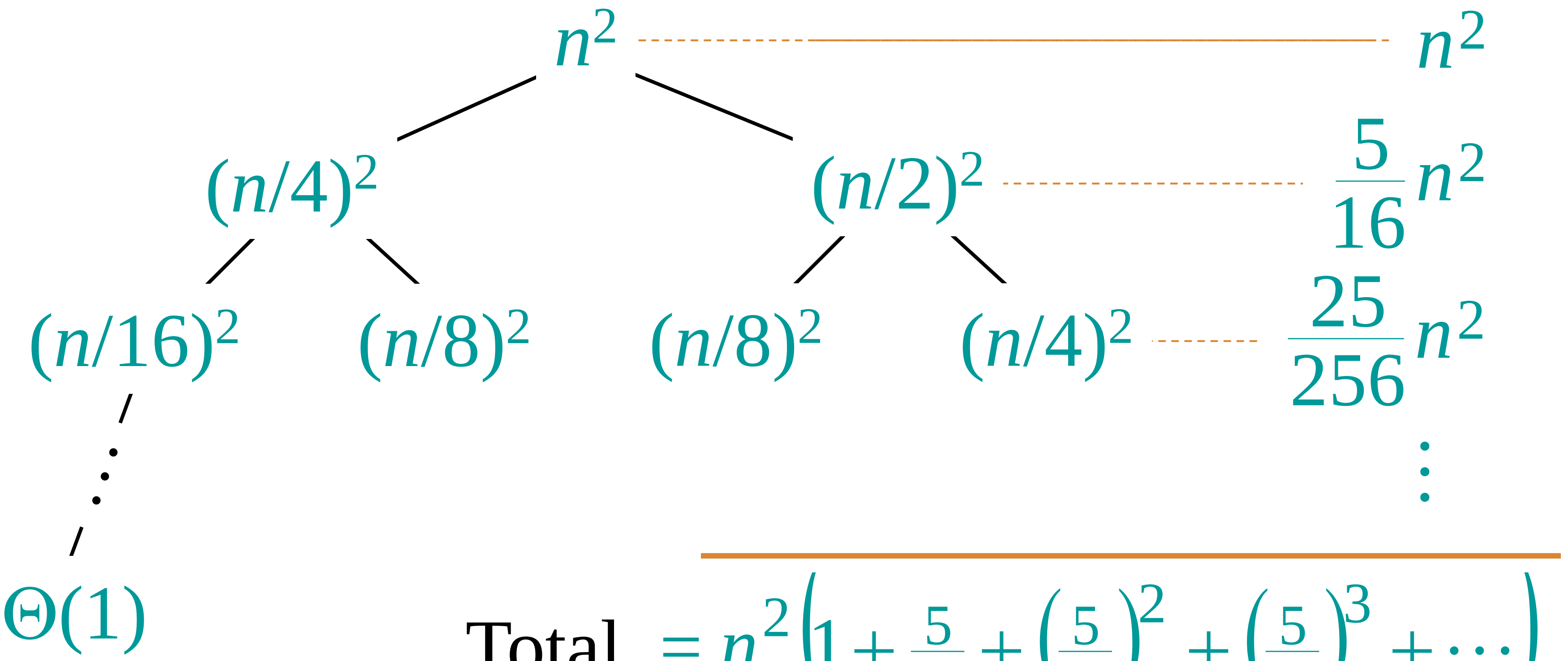
EXAMPLE OF RECURSION TREE

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



EXAMPLE OF RECURSION TREE

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



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

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.

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

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

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

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 $a\,f(n/b) \le c\,f(n)$ for some constant $c < 1$.

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

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

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, $k = 0$.

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

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

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

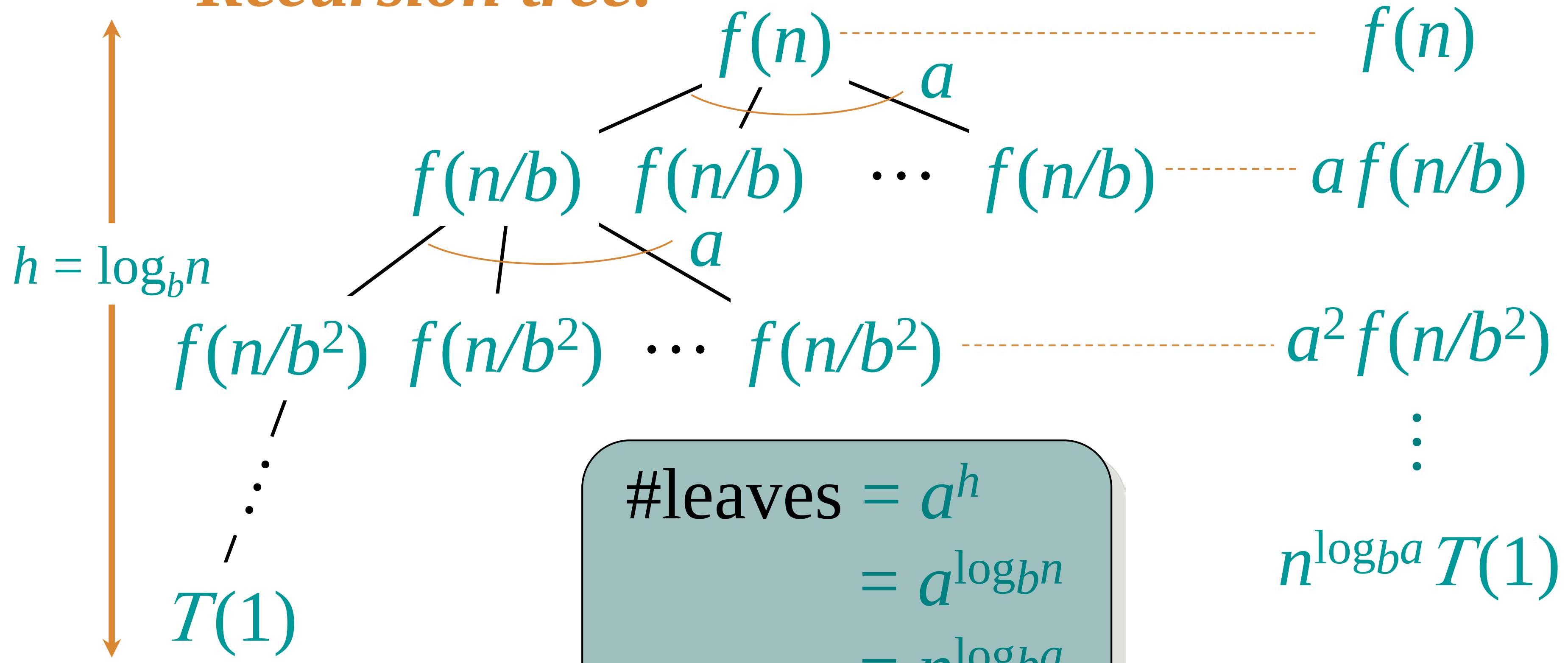
$a = 4,\ b = 2 \Rightarrow n^{\log_b a} = n^2;\ f(n) = n^2/\lg n.$

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

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

IDEA OF MASTER THEOREM

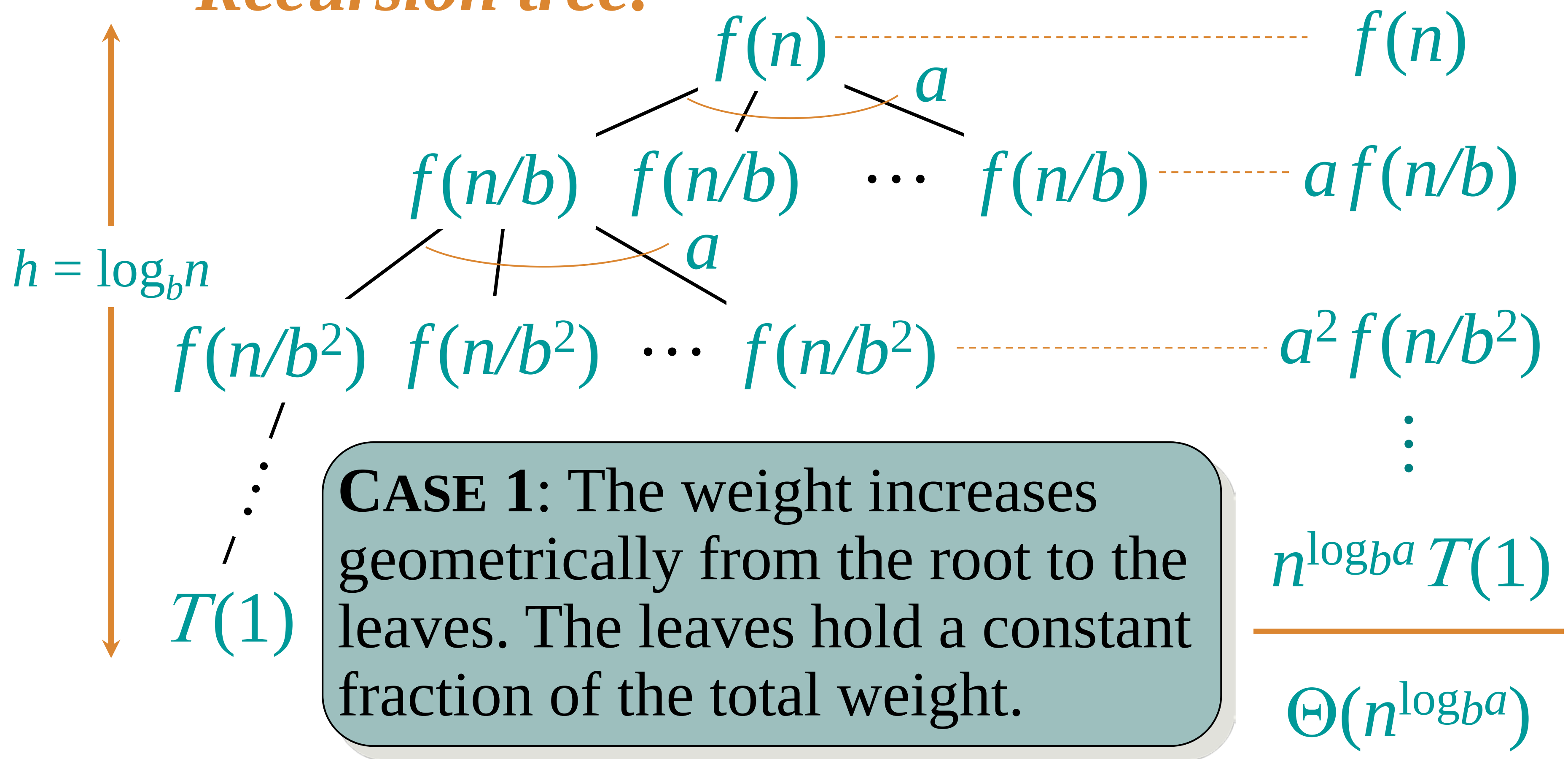
Recursion tree:



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

IDEA OF MASTER THEOREM

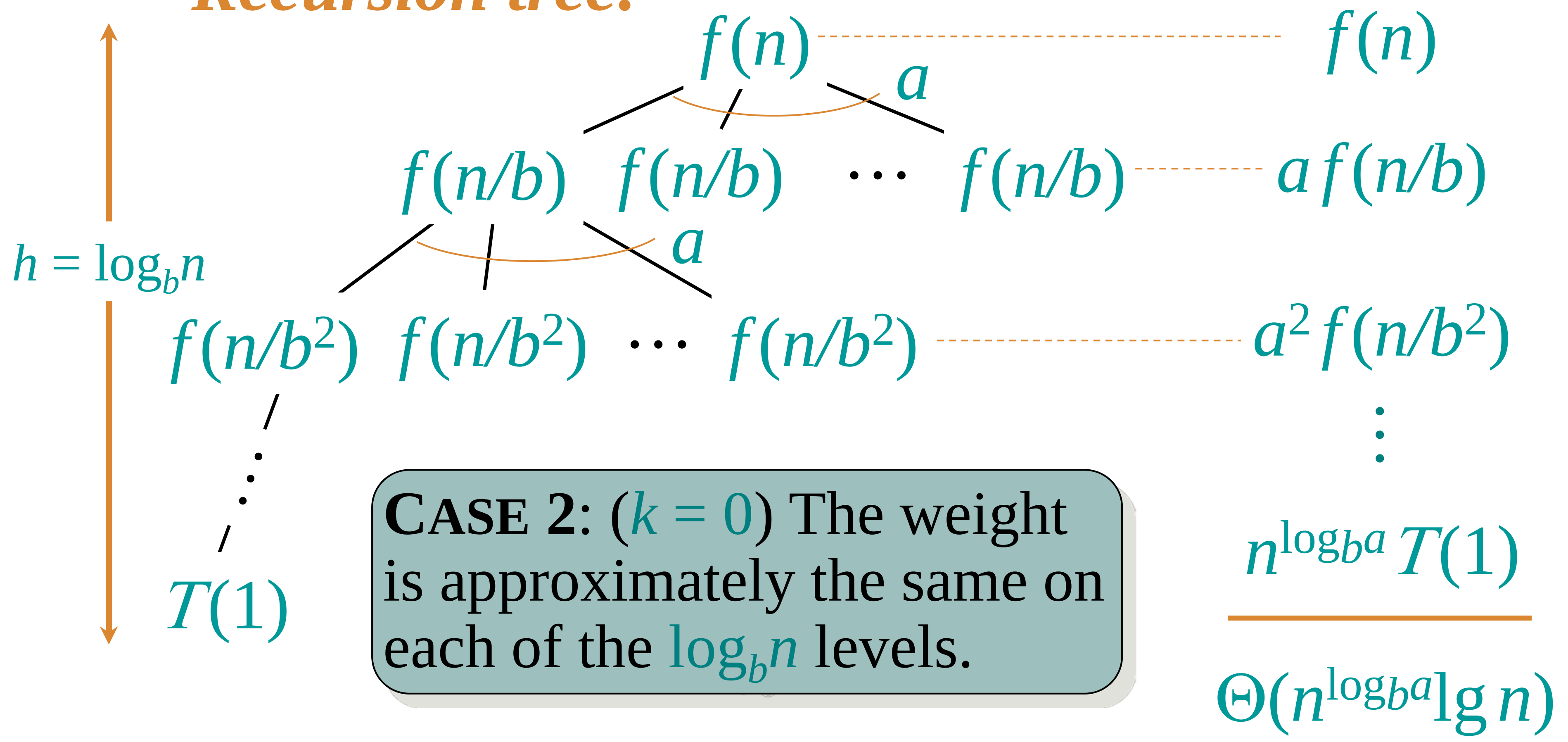
Recursion tree:



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

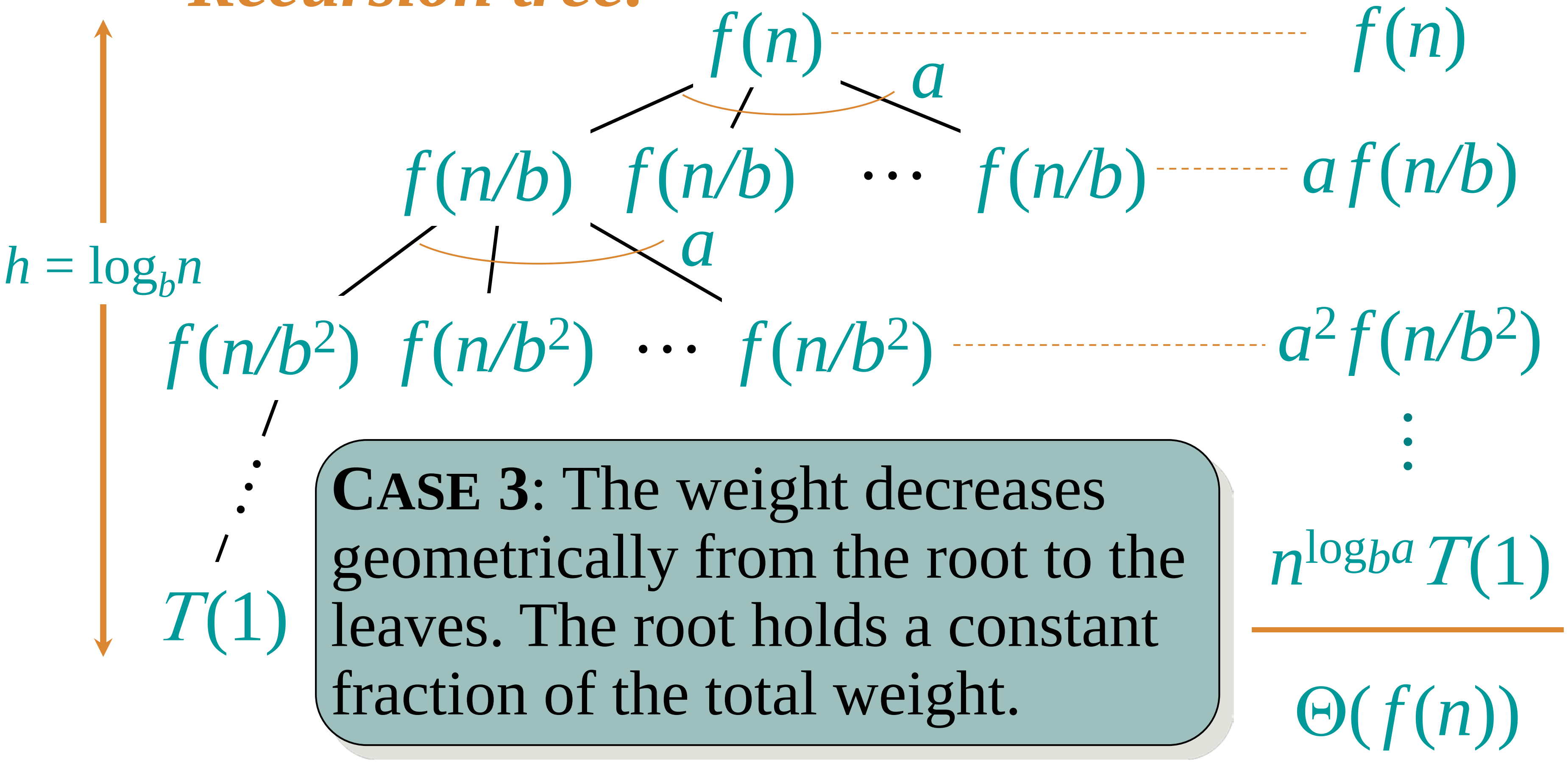
Recursion tree:



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.