Design and Analysis of Algorithms

CSE 5311

Lecture 3 Divide-and-Conquer

Junzhou Huang, Ph.D.

Department of Computer Science and Engineering

Reviewing: Θ-notation

Definition:

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0 \}$

Basic Manipulations:

- Drop low-order terms; ignore leading constants.
- Example: $3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3)$

Reviewing: Insertion Sort Analysis

Worst case: Input reverse sorted.

$$T(n)=\sum_{j=2}^{n}\Theta(j)=\Theta(n^2)$$

[arithmetic series]

Average case: All permutations equally likely.

$$T(n)=\sum_{j=2}^{n}\Theta(j/2)=\Theta(n^2)$$

Is insertion sort a fast sorting algorithm?

- Moderately so, for small n.
- Not at all, for large n.

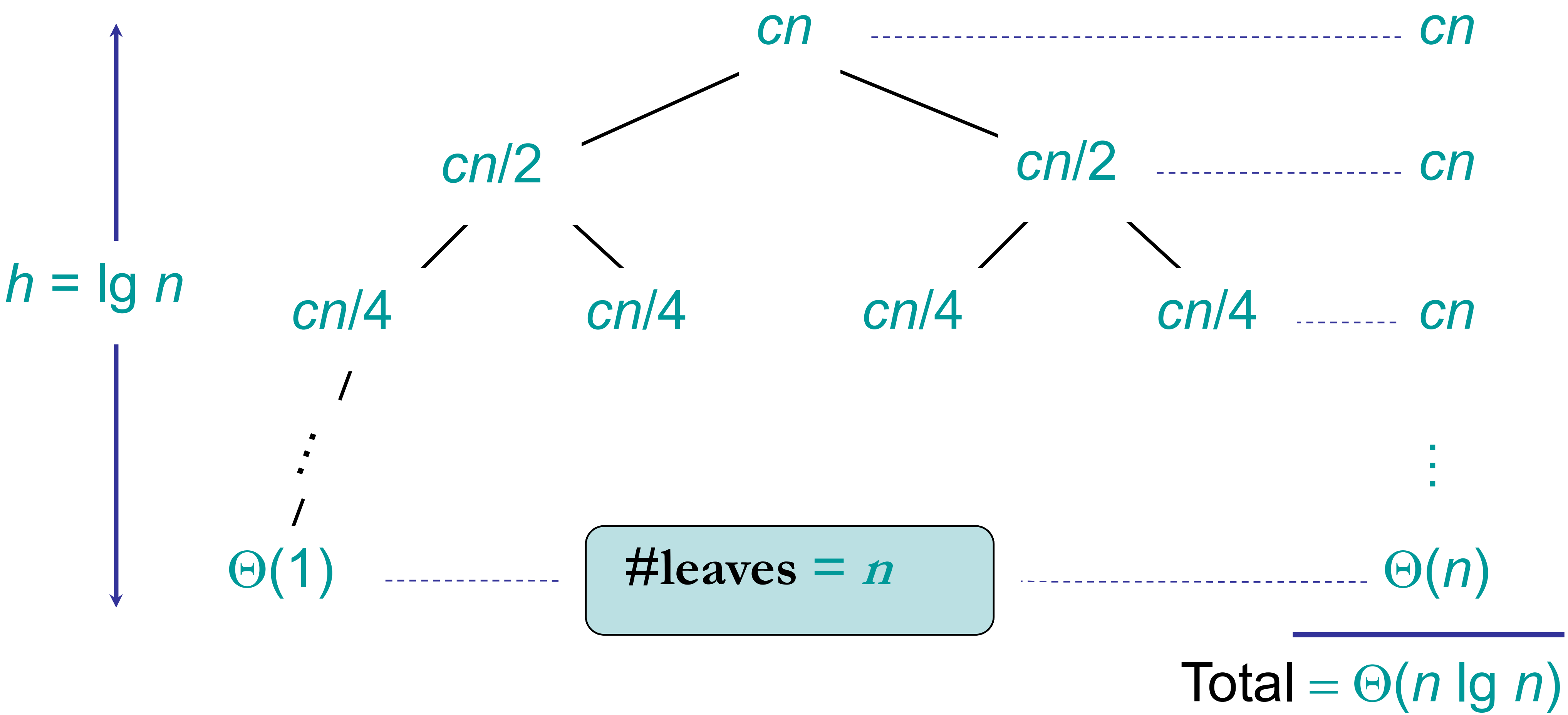
Reviewing: Recurrence for Merge Sort

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

- **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.**
- **Next Lecture will provide several ways to find a good upper bound on $T(n)$.**

Reviewing: Recursion Tree

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



Solving Recurrences

- **Recurrence**
  - The analysis of integer multiplication from last lecture required us to solve a recurrence
  - Recurrences are a major tool for analysis of algorithms
  - Divide and Conquer algorithms which are analyzable by recurrences.

- **Three steps at each level of the recursion:**
  - **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
  - **Conquer** the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
  - **Combine** the solutions to the subproblems into the solution for the original problem.

Recall: Integer Multiplication

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

$$T(n) < 4T(n/2) + \Theta(n)$$

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: $T(n) = 4T(n/2) + \Theta(n)$

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

Example of substitution

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

We can imagine $\Theta(n)=100n$. *Then,* whenever $(c/2)n^3 - 100n \ge 0$, for example, if $c \ge 200$ and $n \ge 1$.

residual

Example

- 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 $1 \leq n < n_0$, we have "$\Theta(1)$" $\leq cn^3$, 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 $T(k) \leq ck^2$ for $k < n$:

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

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 $k < n$.

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

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 (…).
- However, the recursion-tree method promotes intuition

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

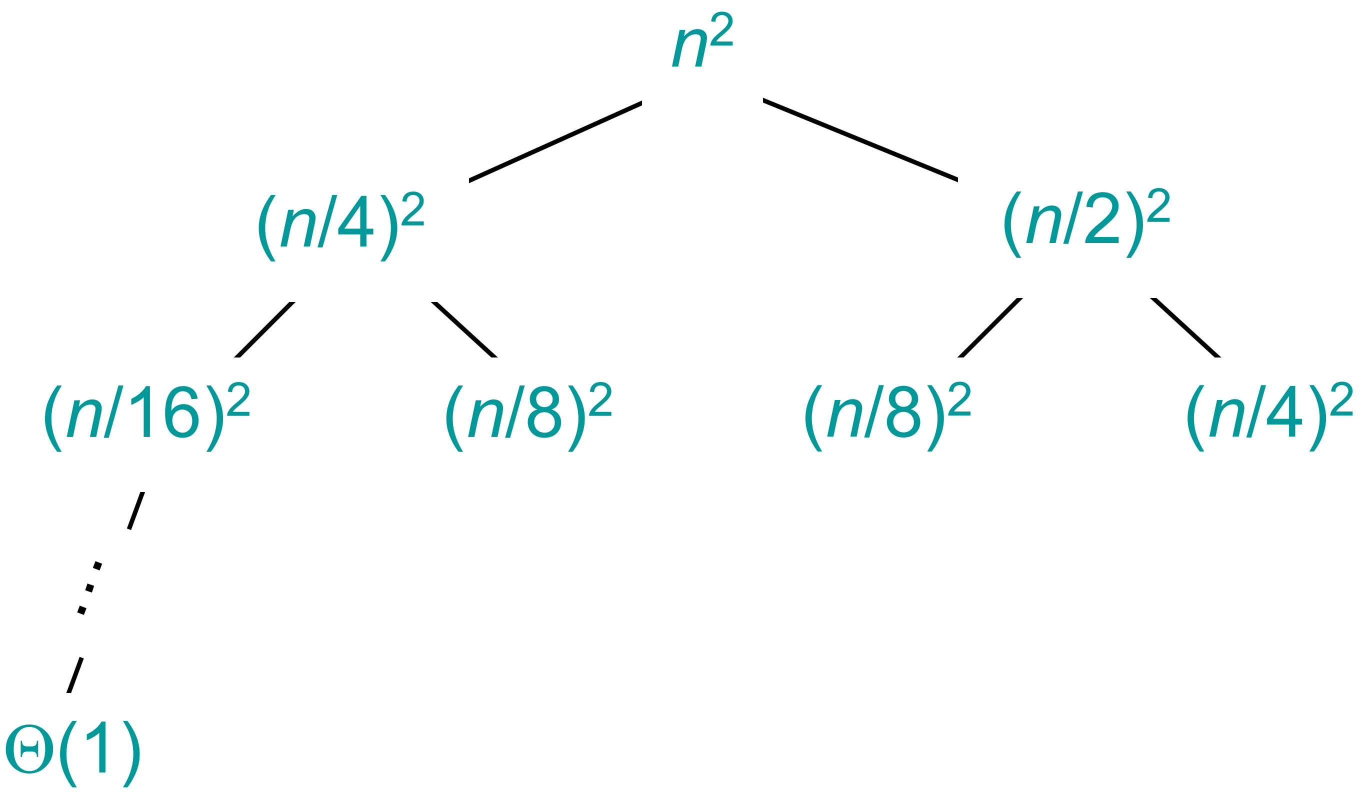
$$n^2$$

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

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

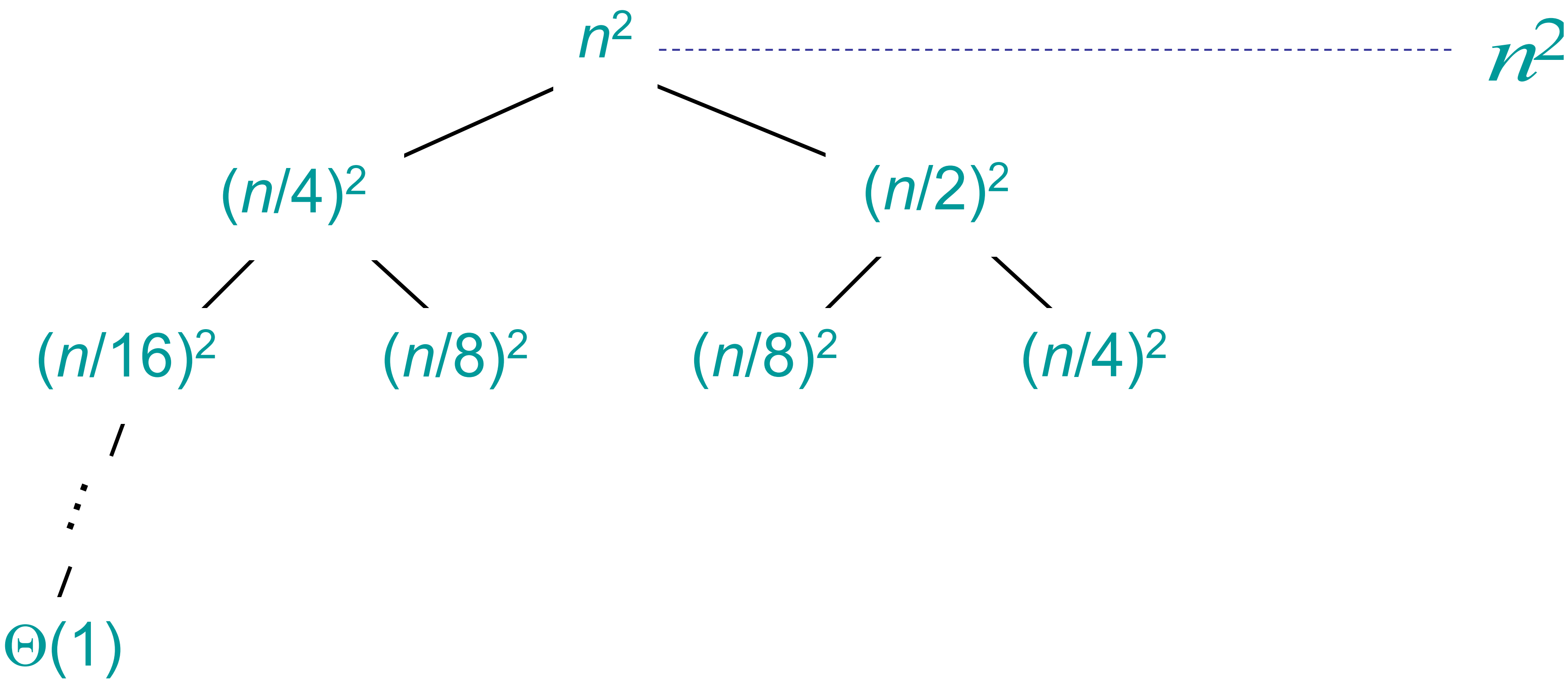
Example of Recursion Tree

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



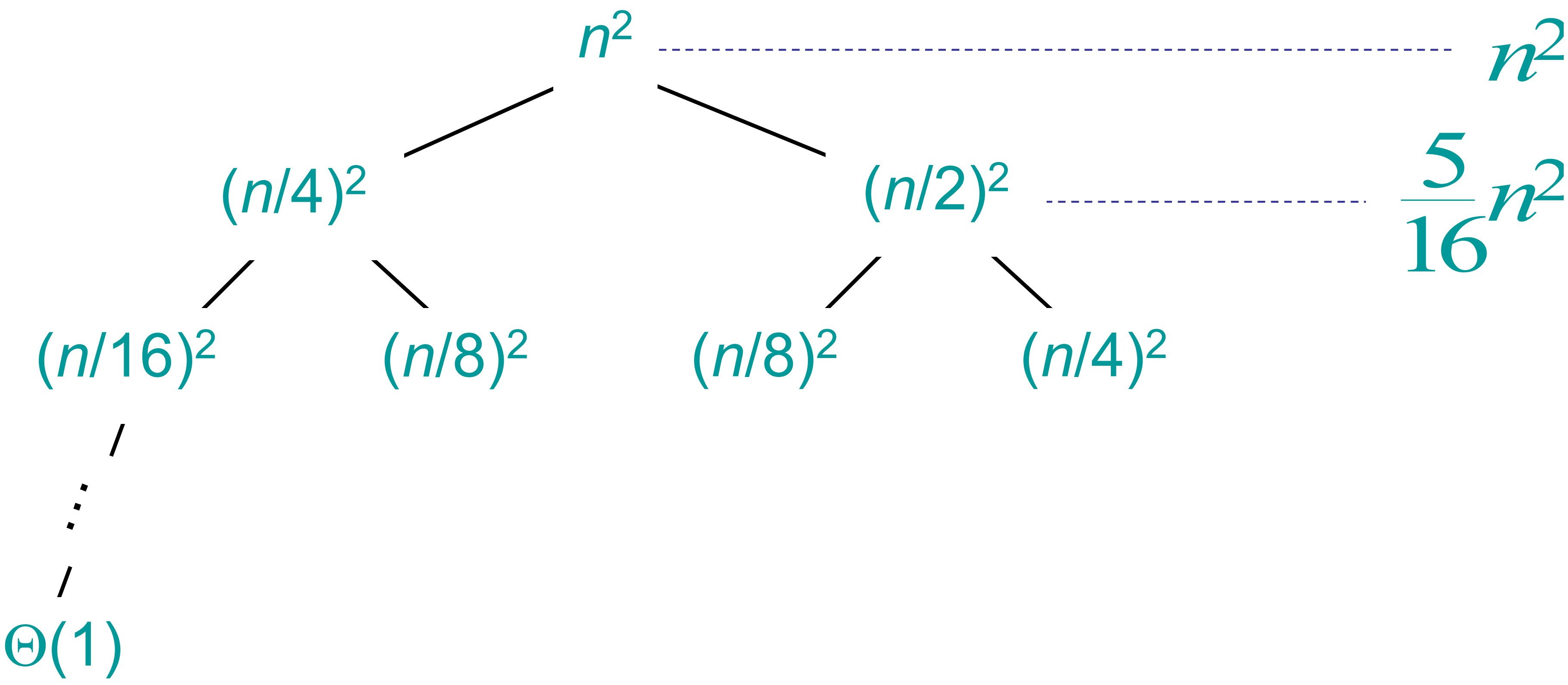
Example of Recursion Tree

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



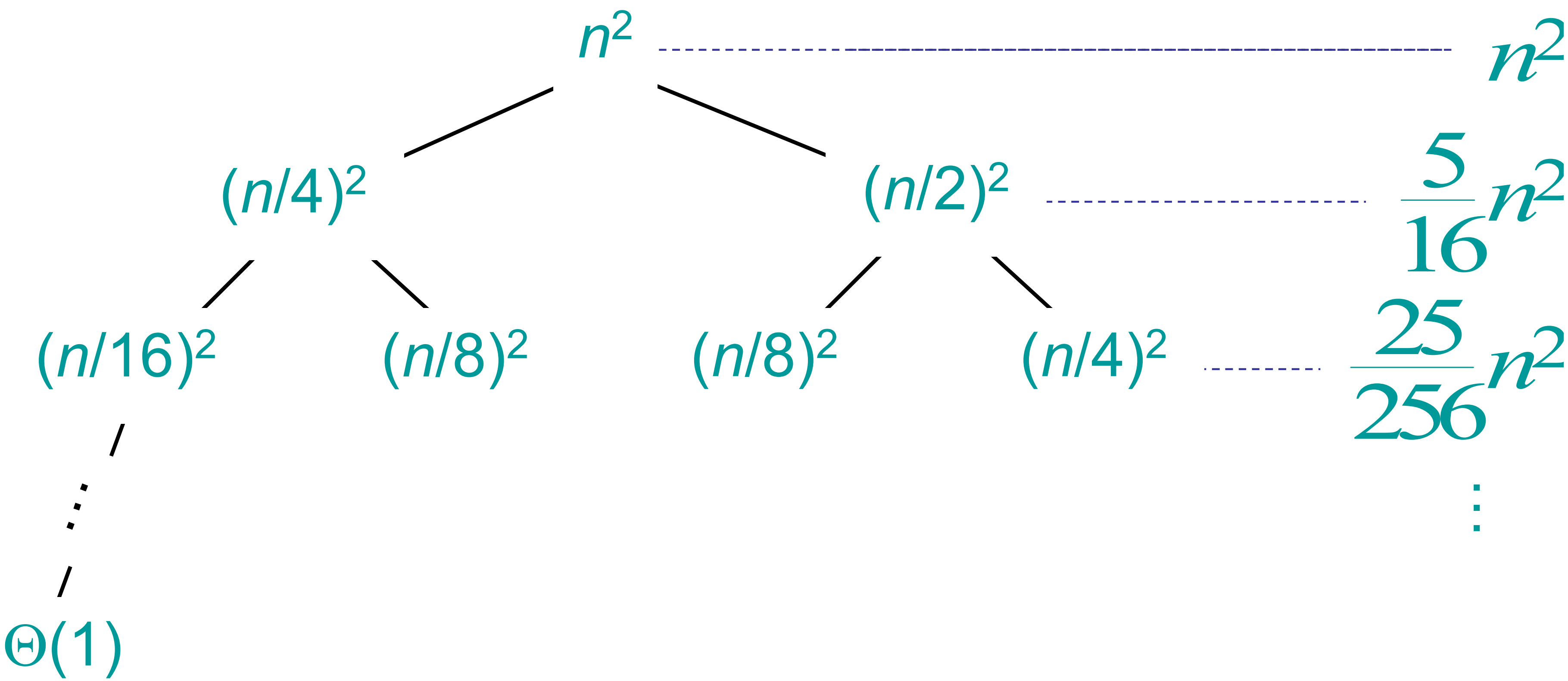
Example of Recursion Tree

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



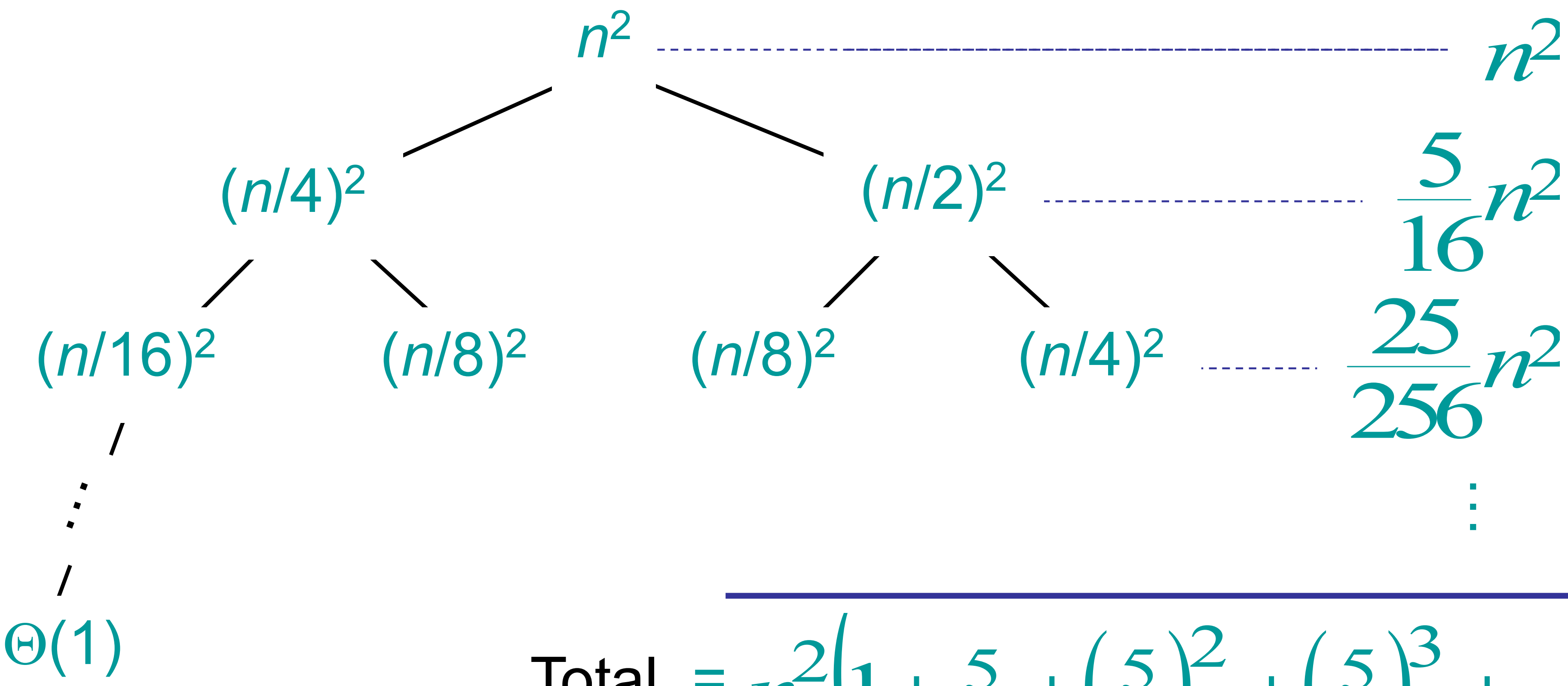
Example of Recursion Tree

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



Example of Recursion Tree

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



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

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

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

The Master Method

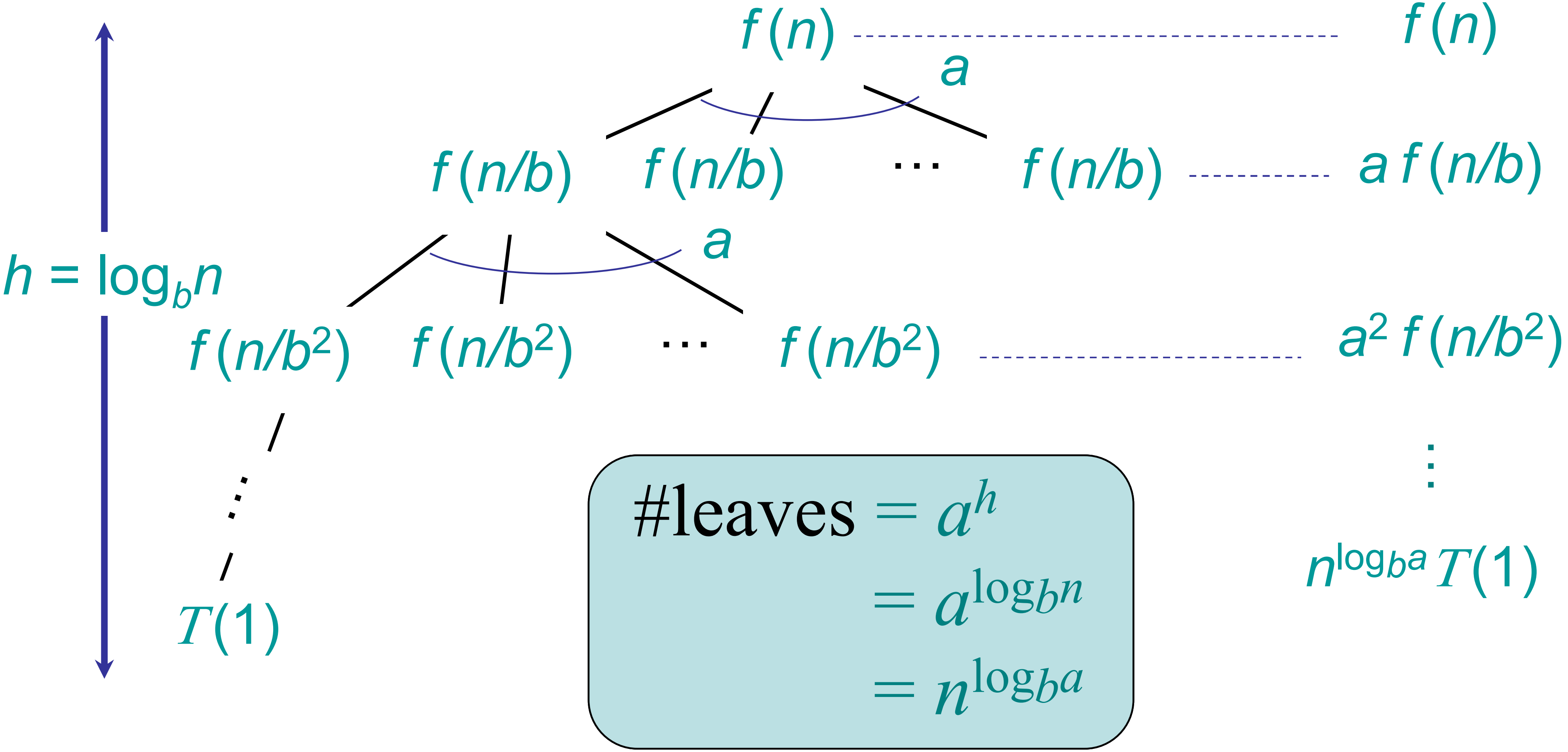
The master method applies to recurrences of the form

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

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

Idea of Master Theorem

Recursion tree:



Case (I)

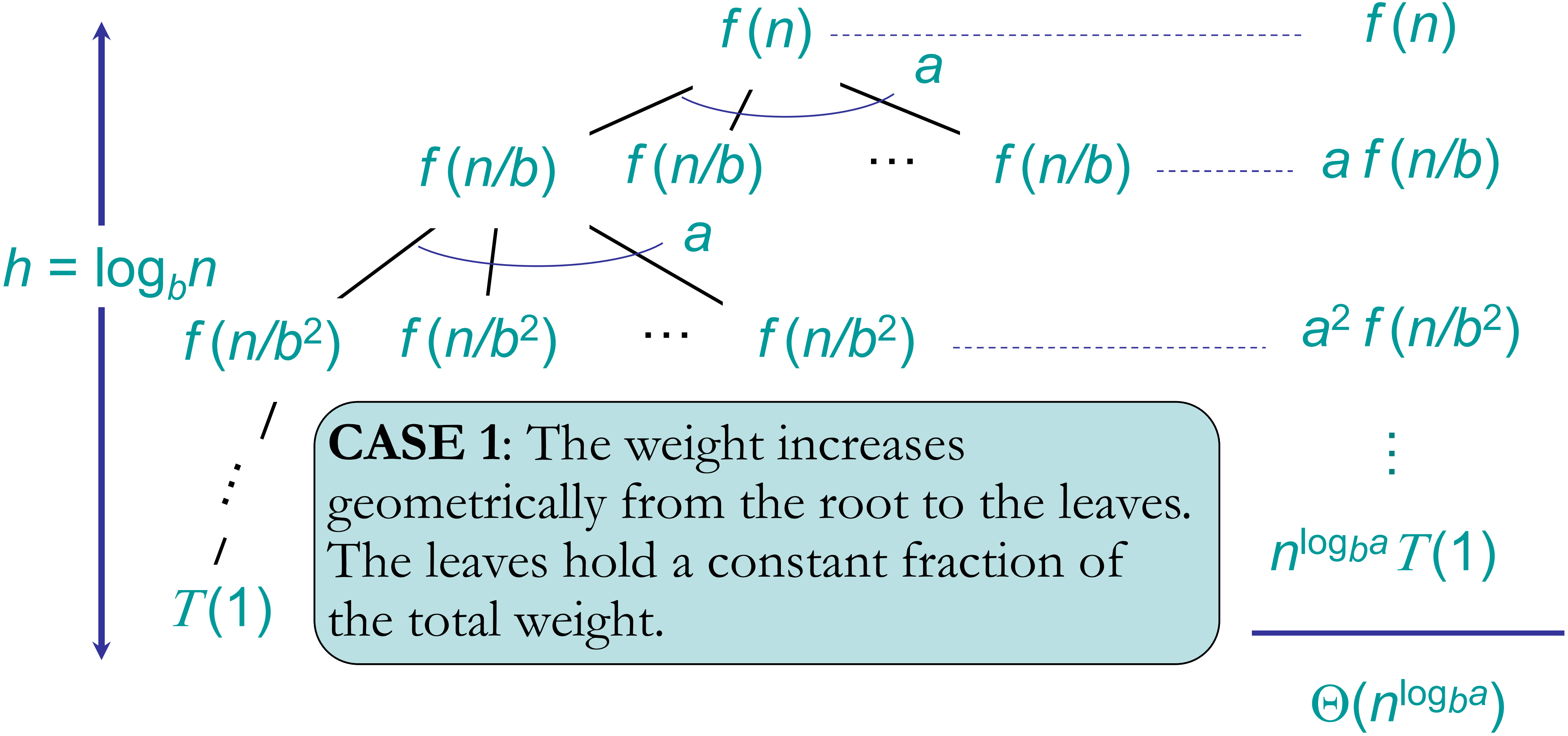
Compare $f(n)$ with $n^{\log_b a}$:

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

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.

$f(n) = n^{\log_b a - \varepsilon}$ and $a\,f(n/b) = a\,(n/b)^{\log_b a - \varepsilon} = b^{\varepsilon}\, n^{\log_b a - \varepsilon}$

Case (II)

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

2. $f(n) = \Theta(n^{\log_b a})$ for some constant $k \geq 0$.
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.

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

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

$h = \log_b n$

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

$\vdots$ $\qquad\qquad \vdots$

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

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

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

$f(n)=n^{\log_b a}$ and $af(n/b)=a\,(n/b)^{\log_b a}= n^{\log_b a}$

Case (III)

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

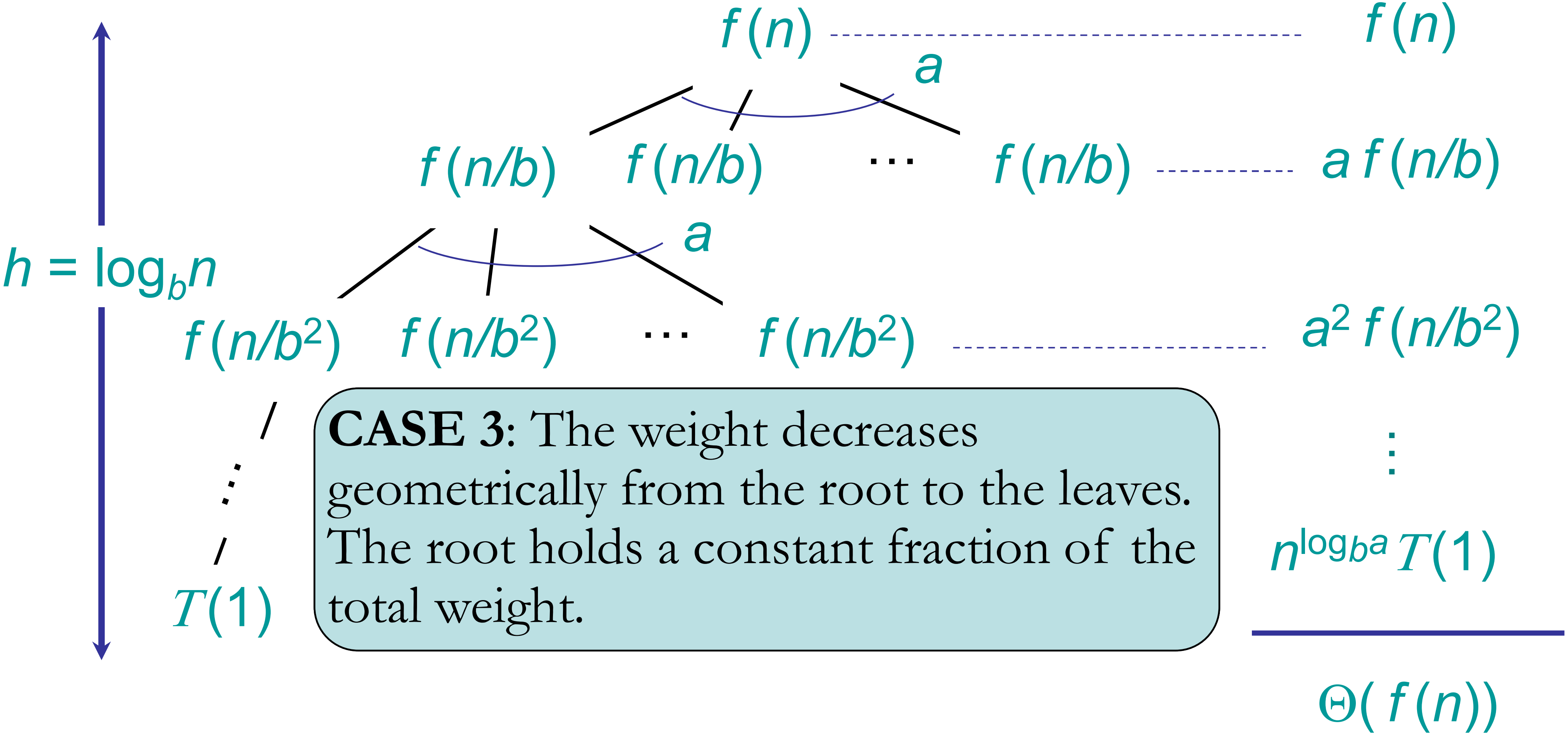
3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^{ε} factor),

   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:



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

$f(n) = n^{\log_b a + \varepsilon}$ and $a\, f(n/b) = a\,(n/b)^{\log_b a + \varepsilon} = b^{-\varepsilon}\, n^{\log_b a + \varepsilon}$

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 $c = ½<1$

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

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 $n^\varepsilon = \omega(\lg n)$.