CS5800 – ALGORITHMS

MODULE 3. DIVIDE & CONQUER - I

Lesson 2: Master Theorem

Ravi Sundaram

Topics

- Statement of the Master Theorem
- Proof of the Master Theorem
- Examples
- Summary

Master Theorem

- Master Theorem is a one-size-fits-all theorem for running times of recursions

Master Theorem: Let **T(n) = a T(n/b) + f (n)** , where $a \geq 1$, $b > 1$, and f is asymptotically positive. Then,

if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$ then $T(n) = \Theta(n^{\log_b a})$

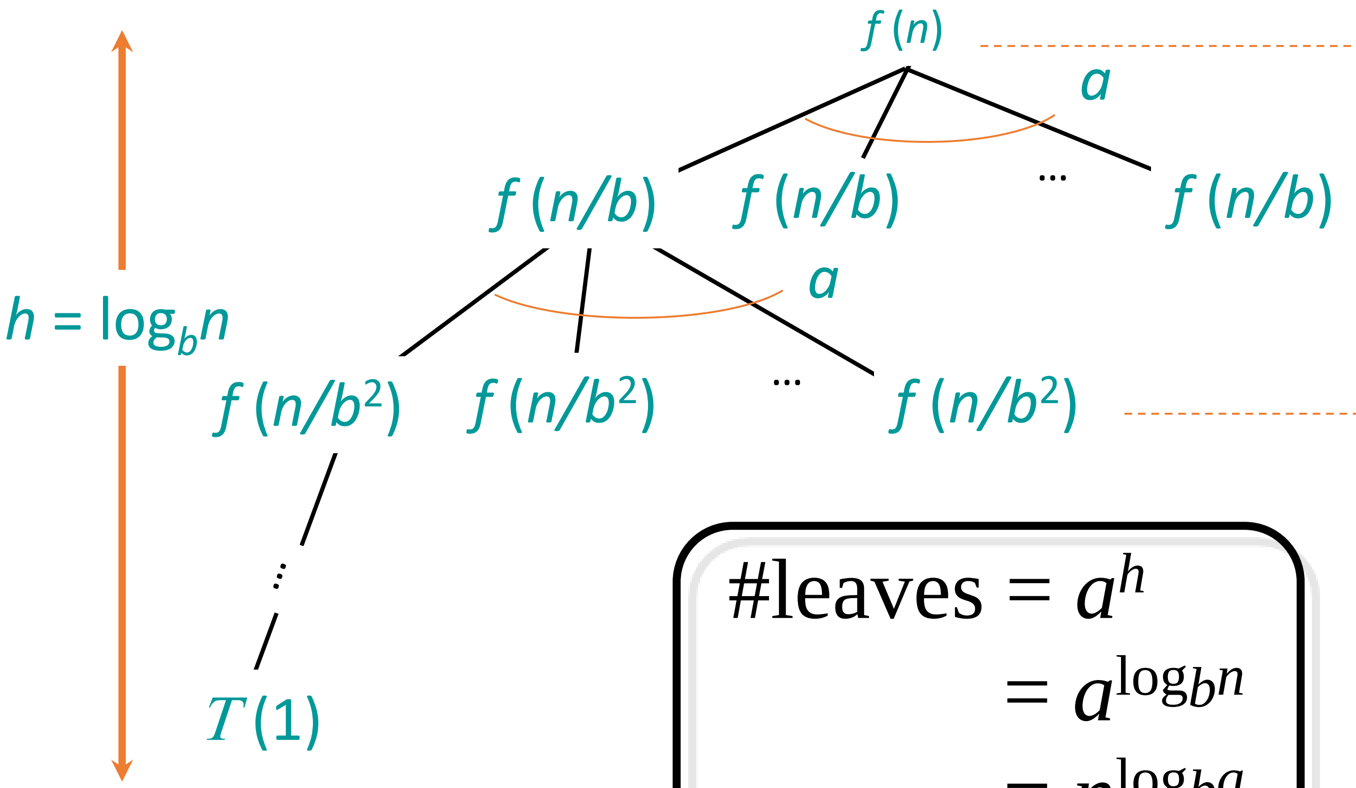
if $f(n) = \Theta(n^{\log_b a} \log^k n)$ for some $k \geq 0$ then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

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

& $af(n/b) \leq cf(n)$ for some $c < 1$ then $T(n) = \Theta(f(n))$

Proof of Master Theorem

Recursion tree:



$f(n)$

$a\,f(n/b)$

$a^2 f(n/b^2)$

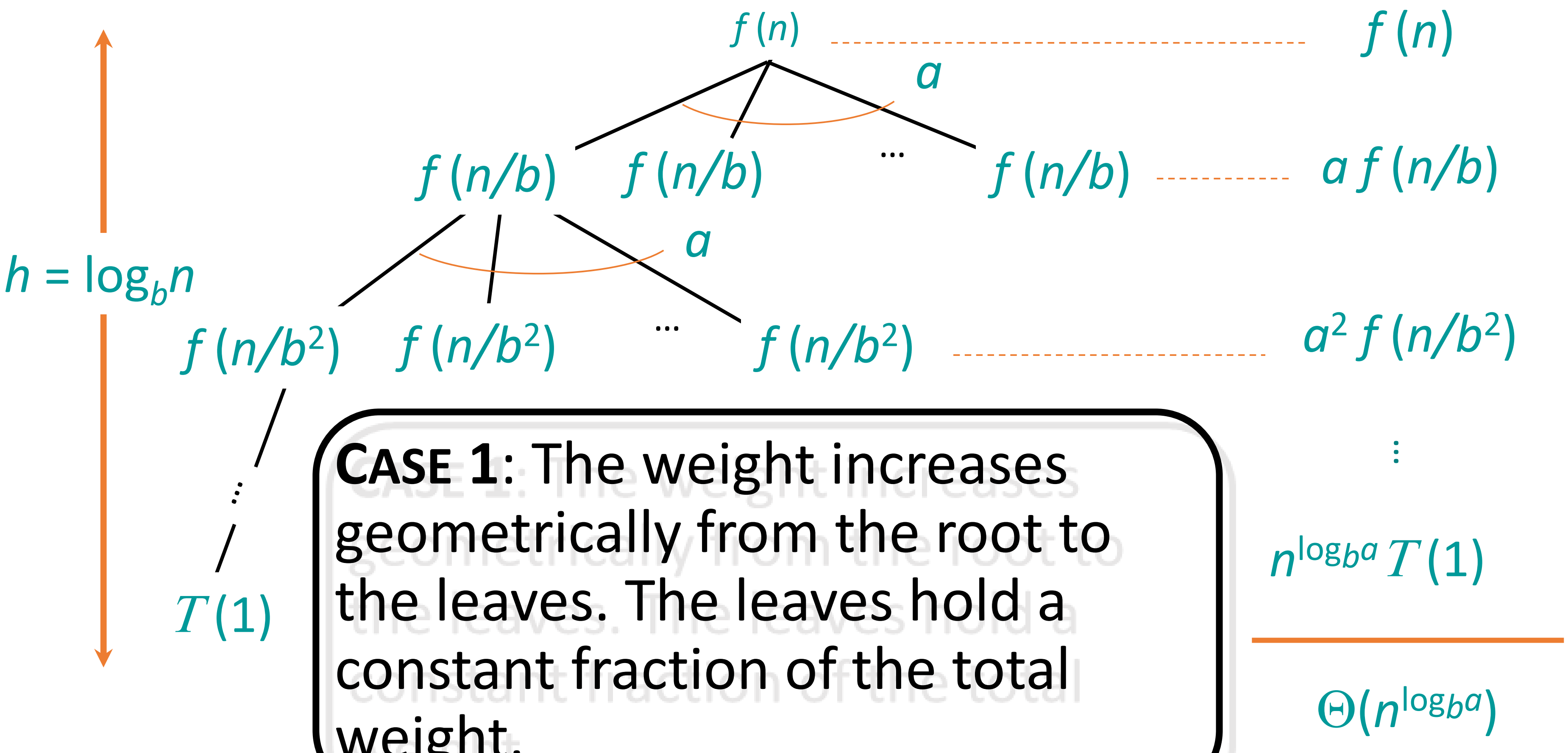
$\vdots$

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

$$\begin{aligned}\#\text{leaves} &= a^h \\ &= a^{\log_b n} \\ &= n^{\log_b a}\end{aligned}$$

Proof 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.

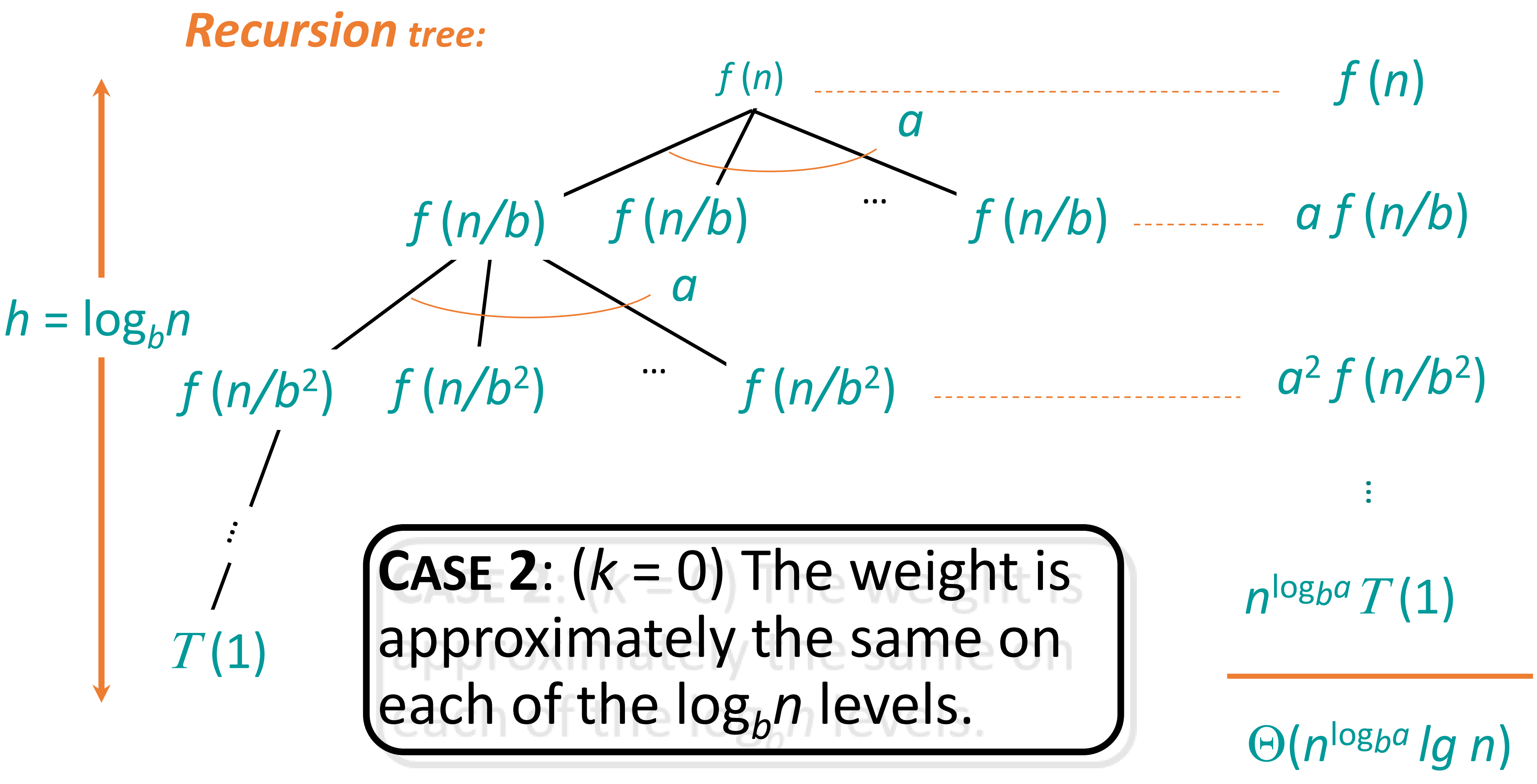
Three common cases

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

Proof of Master Theorem



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

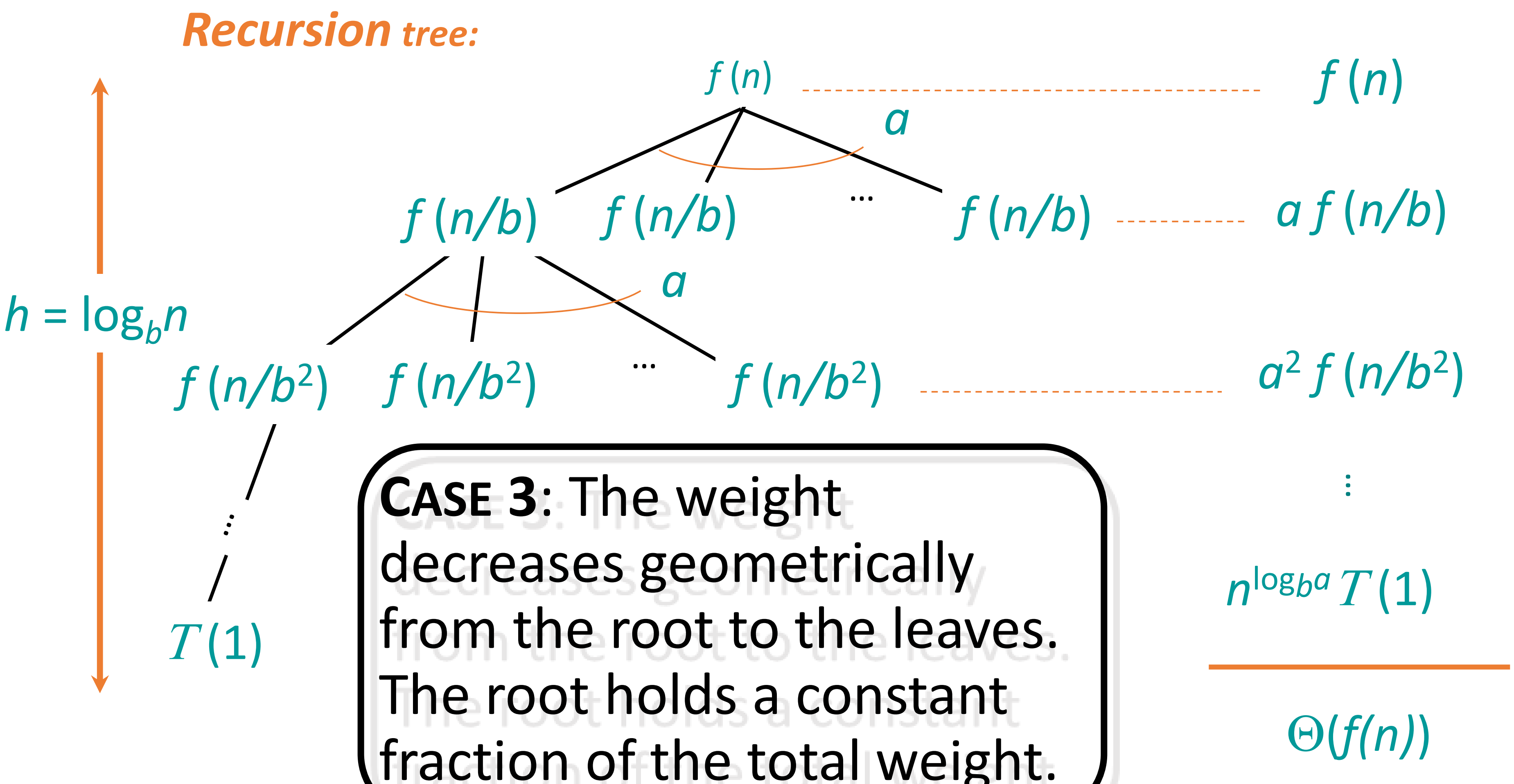
Three common cases

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

Proof of Master Theorem



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

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

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

Summary

- Recursion – useful tool for describing algorithms
- Complexity characterizable using recurrence equation
- Common cases of the Recursion tree method are encapsulated in Master Theorem
- Main Takeaway: To solve $T(n) = aT(n/b) + f(n)$ compare $n^{\log_b a}$ with $f(n)$, whichever dominates is the solution, and if they are equal then throw in an extra log n.