W, 09/04/19

Fall'19 CSCE 629

Analysis of Algorithms

Fang Song
Texas A&M U

Lecture 4

- Exponentiation
- Solving recurrences
  - Recursion tree
  - Master theorem

Logistics

- Announcements
  - HW2 out, due 10am Friday 09/13.
  - Monday: recitation by TA on asymptotics, recurrence, loop invariants; 1-3pm additional office hours by TA HRBB 526
  - Wednesday: guest lecture by Prof. Andreas Klappenecker

Review: Divide-&-Conquer

1. Divide
   - Divide the given instance of the problem into several independent smaller instances of the same problem.
2. Delegate
   - Solve smaller instances recursively, i.e., delegate each smaller instance to the Recursion Fairy.
3. Combine
   - Combine solutions of smaller instance into the final solution for the given instance.

Exponentiation

- Input: integers a, b; b is n-bit long
- Output: $c = a^b$

How many multiplications?

- Naïve algorithm: $\Theta(b) = \Theta(2^n)$
  - Exponential in the input length!
- Divide-and-Conquer
  - Linear in the input length!

1 subproblem
instead of 2 or more

$$a^b = \begin{cases} a^{b/2} \cdot a^{b/2}, & \text{if } b \text{ even} \\ a^{(b-1)/2} \cdot a^{(b-1)/2} \cdot a, & \text{if } b \text{ odd} \end{cases}$$

$$T(b) = T(b/2) + O(1) = O(\log b) = O(n)$$

Recurrence

- Recurrence:
  - Def. an equation or inequality that describes a function in terms of its value on smaller inputs.
  - Sloppiness: ignore floor/ceilings; implicit $T(1) = O(1)$.

$$T(n) = \begin{cases} O(1) \ if \ n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \ \ if \ n > 1 \end{cases}$$

- Recurrences we have seen
  - Merge sort: $T(n) = 2T(n/2) + O(n) = O(n\log n)$
  - Divide-&-Conquer multiplication: $T(n) = 4T(n/2) + O(n) = O(n^2)$
  - Karatsuba's integer multiplication: $T(n) = 3T(n/2) + O(n) \approx O(n^{1.59})$
  - Block-wise matrix multiplication: $T(n) = 8T(n/2) + O(n^2) = O(n^3)$
  - Strassen's matrix multiplication: $T(n) = 7T(n/2) + O(n^2) \approx O(n^{2.81})$
  - Exponentiation: $T(b) = T(b/2) + O(1) = O(\log b) = O(n)$

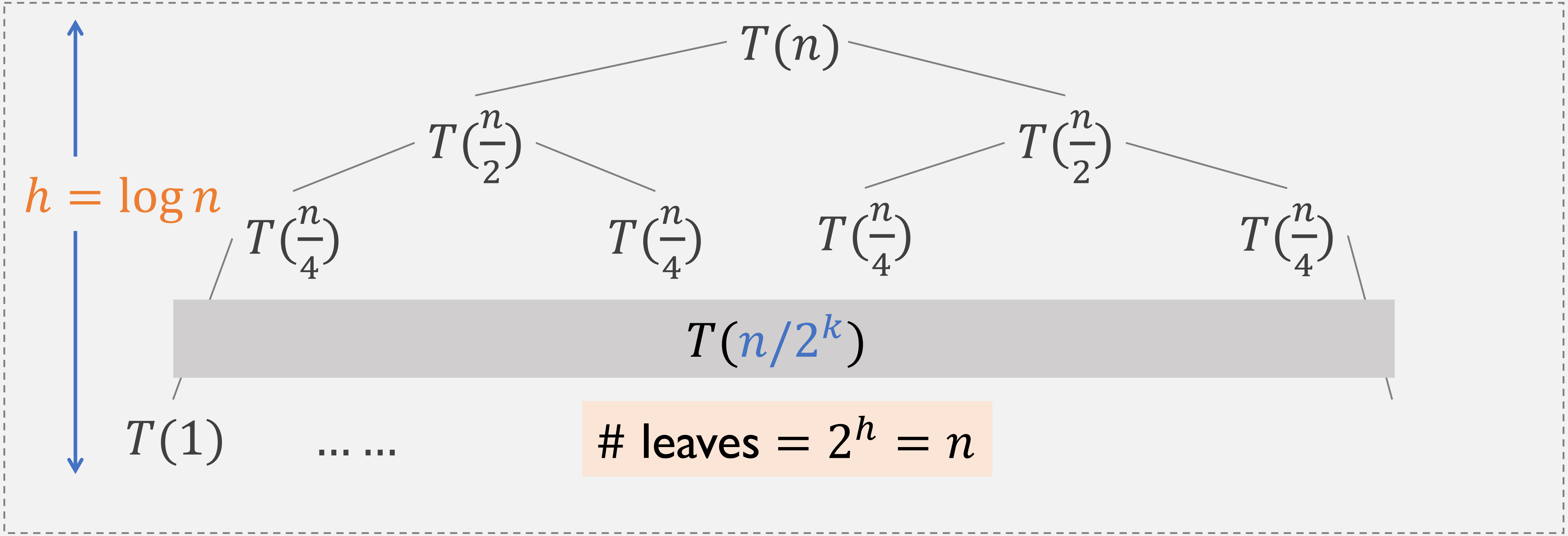
Method #1: Recursion Tree

1. Form recursion tree to guess a solution
   - Draw tree of recursive calls
   - Each node gets assigned the work done during that call to the procedure (dividing and combining)
   - Total work is sum of work at all nodes
2. Prove it by induction

Recursion tree for Mergesort

$T(n) = 2T(n/2) + n$ Ignore floor/ceil & constant factor in merging time $O(n)$

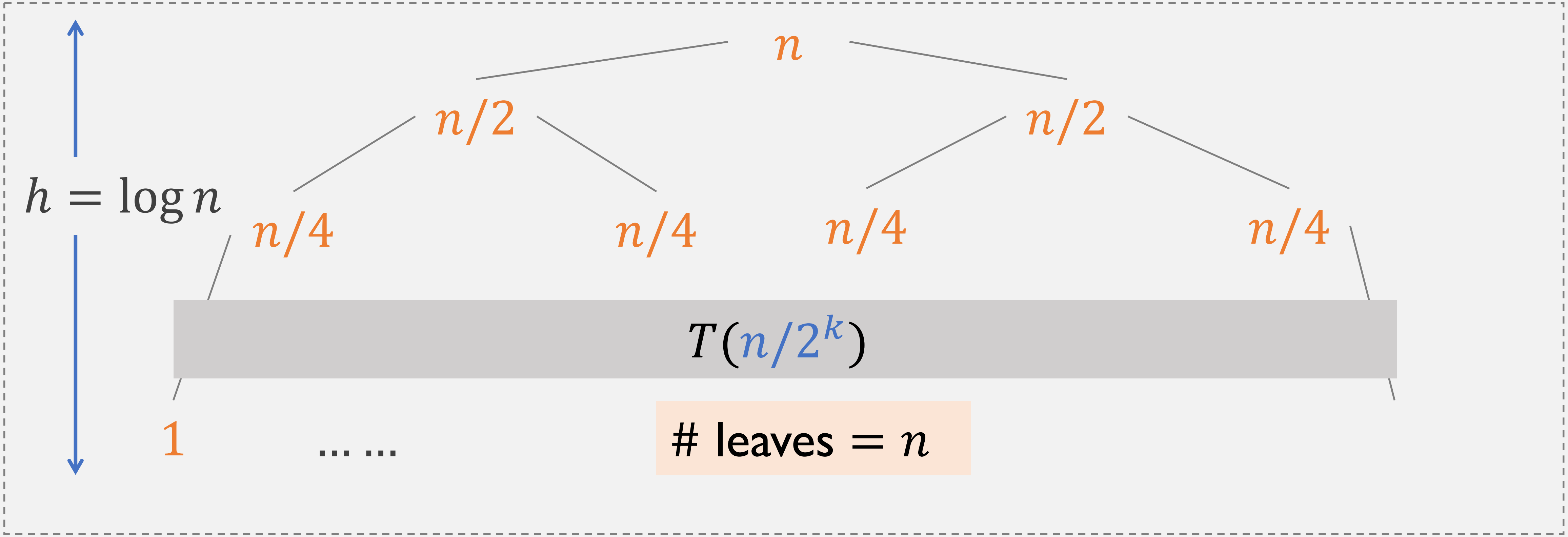
1 Write out tree of recursive calls



Recursion tree for Mergesort

$T(n) = 2T(n/2) + n$ Ignore floor/ceil & constant factor in merging time $O(n)$

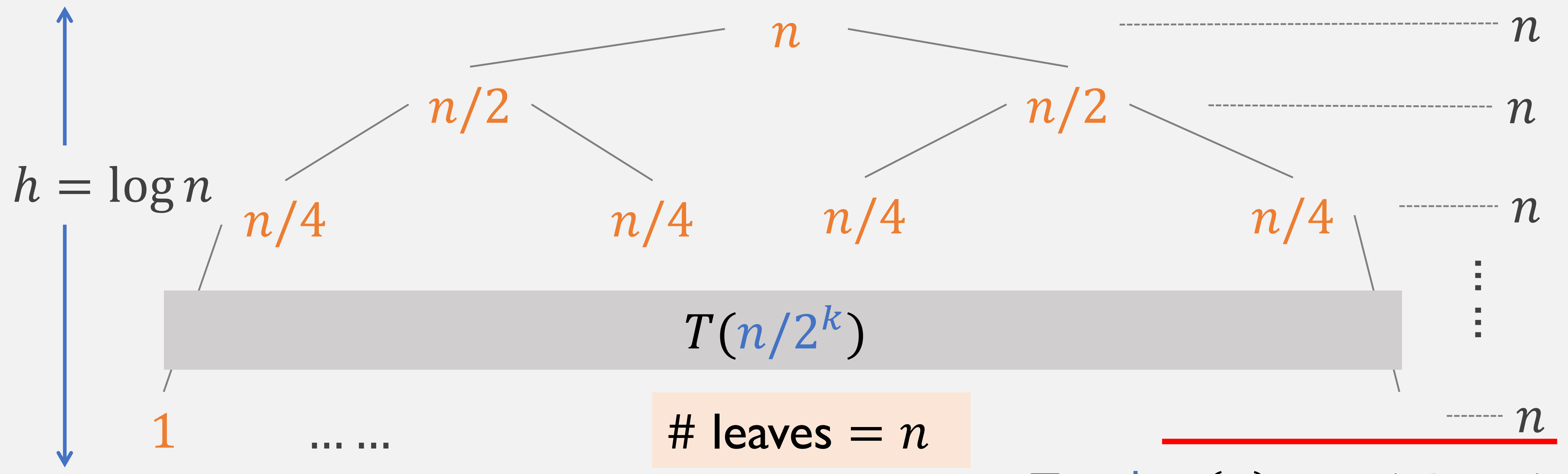
2 Assign work at each level (dividing and combining)



Recursion tree for Mergesort

$T(n) = 2T(n/2) + n$ Ignore floor/ceil & constant factor in merging time $O(n)$

③ Total work = sum of all nodes



Total: $T(n) = O(n\log n)$

Method #2: Master theorem

- A "cookbook" for solving recurrences of the form

$$T(n) = aT(n/b) + f(n)$$

  - $a \geq 1, b > 1$
  - f asymptotically positive, i.e., $f(n) > 0$ for all $n > n_0$

- 3 typical cases depending on $f(n)$ vs. $n^{\log_b a}$

	$T(n)$	$f(n)$ vs. $n^{\log_b a}$
1	$\Theta(n^{\log_b a})$	$f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$
2	$\Theta(n^{\log_b a} \log n)$	$f(n) = O(n^{\log_b a})$
3	$\Theta(f(n))$	$f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ & $f(n/b) \leq cf(n)$ for some $c < 1$

1. $f(n)$ grows polynomially slower by an n^{ϵ} factor

2. Grow at similar rate

3. $f(n)$ grows poly faster + regularity condition

Master theorem in use

1	$\boldsymbol{\Theta(n^{\log_b a})}$	$f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some $\epsilon > 0$
2	$\boldsymbol{\Theta(n^{\log_b a} \log n)}$	$f(n) = O\left(n^{\log_b a}\right)$
3	$\boldsymbol{\Theta(f(n))}$	$f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some $\epsilon > 0$ & $f(n/b) \le c\, f(n)$ for some $c < 1$

- In-class exercise: solve these by master theorem
    1. Merge sort: $T(n) = 2T(n/2) + O(n)$
    2. Divide-&-Conquer multiplication: $T(n) = 4T(n/2) + O(n)$
    3. Karatsuba's integer multiplication: $T(n) = 3T(n/2) + O(n)$
    4. Block-wise matrix multiplication: $T(n) = 8T(n/2) + O(n^2)$
    5. Strassen's matrix multiplication: $T(n) = 7T(n/2) + O(n^2)$
    6. Exponentiation: $T(b) = T(b/2) + O(1)$
    7. $T(n) = 4T(n/2) + O(n^3)$

Master theorem in use

1	$\boldsymbol{\Theta(n^{\log_b a})}$	$f(n) = O\left(n^{\log_b a-\epsilon}\right)$ for some $\epsilon > 0$
2	$\boldsymbol{\Theta(n^{\log_b a}\log n)}$	$f(n) = O\left(n^{\log_b a}\right)$
3	$\boldsymbol{\Theta(f(n))}$	$f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ for some $\epsilon > 0$ & $f(n/b) \leq c\, f(n)$ for some $c < 1$

- Ex1. $T(n) = 2T(n/2) + O(n)$ [Mergesort]
  - $a = 2, b = 2, n^{\log_b a} = n = f(n)$
  - Case 2 $\Rightarrow T(n) = O(n\log n)$

Master theorem in use

1	$\boldsymbol{\Theta(n^{\log_b a})}$	$f(n) = O\left(n^{\log_b a-\epsilon}\right)$ for some $\epsilon > 0$
2	$\boldsymbol{\Theta(n^{\log_b a}\log n)}$	$f(n) = O\left(n^{\log_b a}\right)$
3	$\boldsymbol{\Theta(f(n))}$	$f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ for some $\epsilon > 0$ & $f(n/b) \le c\, f(n)$ for some $c < 1$

- Ex2. $T(n) = 4T(n/2) + O(n)$ [Divide-&-Conquer mult.]
  - $a = 4, b = 2, n^{\log_b a} = n^2, f(n) = n = O(n^{2-\epsilon})$ (pick $\epsilon = 1$)
  - Case 1 $\Rightarrow T(n) = O(n^2)$
- Ex3. $T(n) = 3T(n/2) + O(n)$ [Karatsuba's integer mult.]
  - $a = 3, b = 2, n^{\log_b a} = n^{\log 3} \approx n^{1.58}, f(n) = n = O(n^{1.58-\epsilon})$ for $\epsilon = .5$
  - Case 1 $\Rightarrow T(n) = O(n^{\log 3})$

Master theorem in use

1	$\boldsymbol{\Theta(n^{\log_b a})}$	$f(n) = O\left(n^{\log_b a-\epsilon}\right)$ for some $\epsilon > 0$
2	$\boldsymbol{\Theta(n^{\log_b a} \log n)}$	$f(n) = O\left(n^{\log_b a}\right)$
3	$\boldsymbol{\Theta(f(n))}$	$f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ for some $\epsilon > 0$ & $f(n/b) \leq c\, f(n)$ for some $c < 1$

- Ex4. : $T(n) = 8T(n/2) + O(n^2)$ [Block-wise matrix mult.]
  - $a = 8, b = 2, n^{\log_b a} = n^{\log_2 8} = n^3, f(n) = n^2 = O(n^{3-\epsilon})$ for $\epsilon = 1$
  - Case 1 $\Rightarrow T(n) = O\left(n^{\log 8}\right) = O(n^3)$
- Ex5. $T(n) = 7T(n/2) + O(n^2)$ [Strassen's matrix mult.]
  - $a = 7, b = 2, n^{\log_b a} = n^{\log_2 7} \approx n^{2.81}, f(n) = n^2 = O(n^{2.81-\epsilon})$ for $\epsilon = .8$
  - Case 1 $\Rightarrow T(n) = O\left(n^{\log 7}\right) \approx O(n^{2.81})$

Master theorem in use

1	$\boldsymbol{\Theta(n^{\log_b a})}$	$f(n) = O\left(n^{\log_b a-\epsilon}\right)$ for some $\epsilon > 0$
2	$\boldsymbol{\Theta(n^{\log_b a}\log n)}$	$f(n) = O\left(n^{\log_b a}\right)$
3	$\boldsymbol{\Theta(f(n))}$	$f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ for some $\epsilon > 0$ & $f(n/b) \leq c\, f(n)$ for some $c < 1$

- Ex6. $T(n) = T(n/2) + O(1)$ [Exponentiation]
  - $a = 1, b = 2, n^{\log_b a} = n^0 = 1, f(n) = O(1)$
  - Case 2 $\Rightarrow T(n) = O(n^0 \log n) = O(\log n)$
- Ex7. $T(n) = 4T(n/2) + O(n^3)$
  - $a = 4, b = 2, n^{\log_b a} = n^2, f(n) = n^3 = \Omega(n^{2+\epsilon})$ for $\epsilon = 1$
  - Case 3 $\Rightarrow T(n) = \Theta(n^3)$
  - Don't forget to check the regularity condition: $(n/2)^3 \leq c\, n^3$ for $c = .5 < 1$

Master theorem in use

1	$\boldsymbol{\Theta(n^{\log_b a})}$	$f(n) = O\left(n^{\log_b a-\epsilon}\right)$ for some $\epsilon > 0$
2	$\boldsymbol{\Theta(n^{\log_b a}\log n)}$	$f(n) = O\left(n^{\log_b a}\right)$
3	$\boldsymbol{\Theta(f(n))}$	$f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ for some $\epsilon > 0$ $f(n/b) \leq c\, f(n)$ for some $c < 1$

- Ex8. $T(n) = 4T(n/2) + n^2/\log n$
  - $a = 4, b = 2, n^{\log_b a} = n^2, f(n) = n^2/\log n$
  - Master theorem doesn't apply! For any constant $\epsilon > 0, n^\epsilon = \omega(\log n)$

- Generalization exists
  - Ex. Akra–Bazzi method
    https://en.wikipedia.org/wiki/Akra%E2%80%93Bazzi_method

Master theorem: proof idea

$h = \log_b n$

$f(n)$ — a branches — $f(n)$

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

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

$\vdots$

Let $f(n) = n^d$. 3 cases: $f(n)$ vs. $n^{\log_b a} \Leftrightarrow d$ vs. $\log_b a$

$$T(n) = n^d + a(n/b)^d + a^2(n/b^2)^d + \cdots = \sum_k f(n) \cdot \left(a/b^d\right)^k$$

$T(1)$

width (# leaves) = — $n^{\log_b a}$

$a^h = a^{\log_b n} = n^{\log_b a}$

Master theorem: proof idea

$f(n)$ a branches — $f(n)$

a $f(n/b)$ $f(n/b)$ $\cdots$ $f(n/b)$ — $af(n/b)$

$h = \log_b n$

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

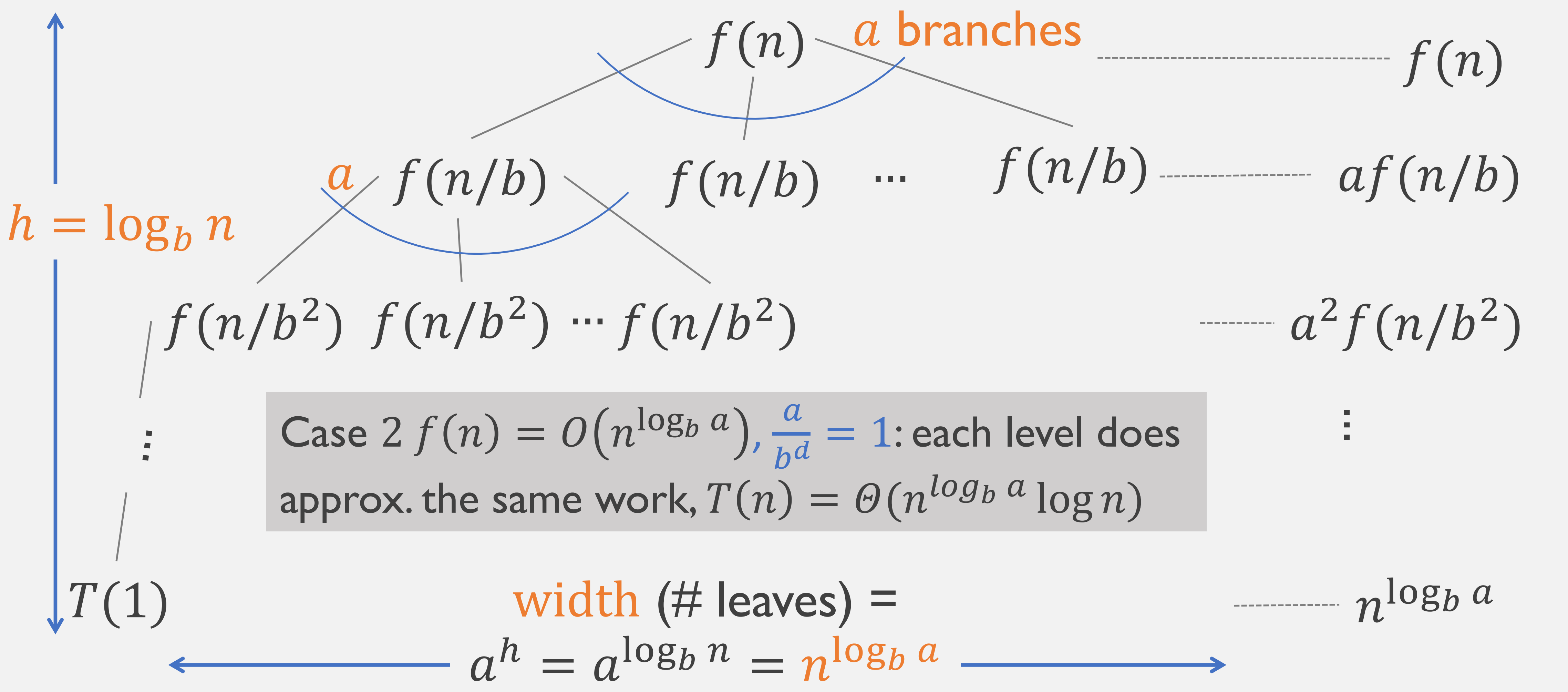
$\vdots$

Case 1 $f(n) = O\left(n^{\log_b a - \epsilon}\right)$, i.e.,$d < \log_b a$, $\frac{a}{b^d} > 1$:
weight increases geometrically from root to leaves.
Leaves dominate, $T(n) = \Theta(n^{\log_b a})$

$T(1)$ width (# leaves) = — $n^{\log_b a}$

$a^h = a^{\log_b n} = n^{\log_b a}$

Master theorem: proof idea



Case 2 $f(n) = O\left(n^{\log_b a}\right)$, $\frac{a}{b^d} = 1$: each level does approx. the same work, $T(n) = \Theta(n^{\log_b a} \log n)$

Master theorem: proof idea

$f(n)$ a branches — $f(n)$

a $f(n/b)$ $f(n/b)$ $\cdots$ $f(n/b)$ — $af(n/b)$

$h = \log_b n$

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

$\vdots$ $\vdots$

Case 3 $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right), \frac{a}{b^d} < 1$: weight decreases geometrically from root to leave.
Root dominates, $T(n) = \Theta(f(n))$

$T(1)$

width (# leaves) = $a^h = a^{\log_b n} = n^{\log_b a}$ — $n^{\log_b a}$