CS4102 Algorithms
Spring 2019

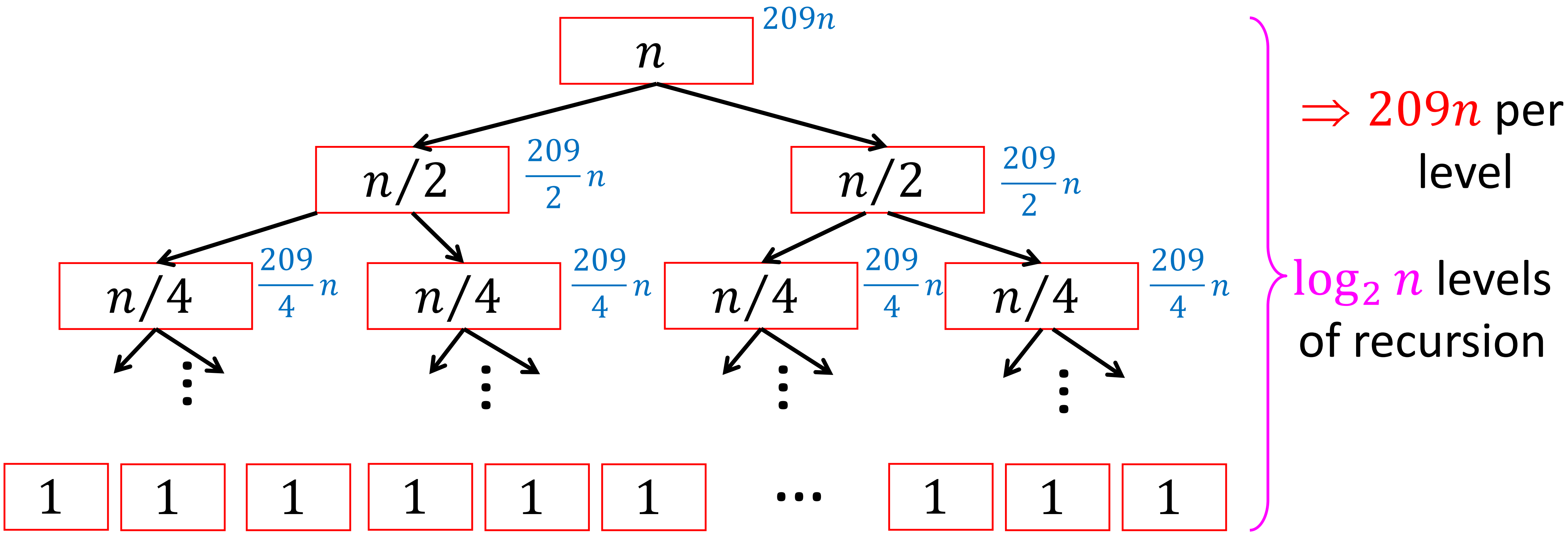
Warm Up

What is the asymptotic run time of MergeSort if its recurrence is

$$T(n) = 2T\left(\frac{n}{2}\right) + 209n$$

Tree method

$$T(n) = 2T(\frac{n}{2}) + 209n$$



$$T(n) = 209 \sum_{i=1}^{\log n} n = 209n \log_2 n$$

Today's Keywords

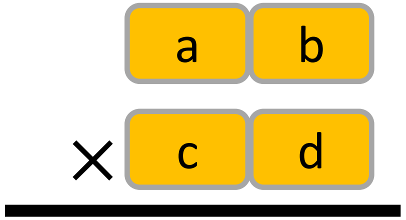
- Karatsuba (finishing up)
- Guess and Check Method
- Induction
- Master Theorem

CLRS Readings

- Chapter 4

Homeworks

- Hw1 due ~~Wed, January 30 at 11pm~~ Sunday, Feb 3 at 11pm
  - Start early!
  - Written (use Latex!) – Submit BOTH pdf and zip!
  - Asymptotic notation
  - Recurrences
  - Divide and Conquer



Karatsuba Algorithm

1. Recursively compute: $ac, bd, (a+b)(c+d)$
2. $(ad+bc) = (a+b)(c+d) - ac - bd$
3. Return $10^n(ac) + 10^{\frac{n}{2}}(ad+bc) + bd$

$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

<u>Pseudo-code</u>

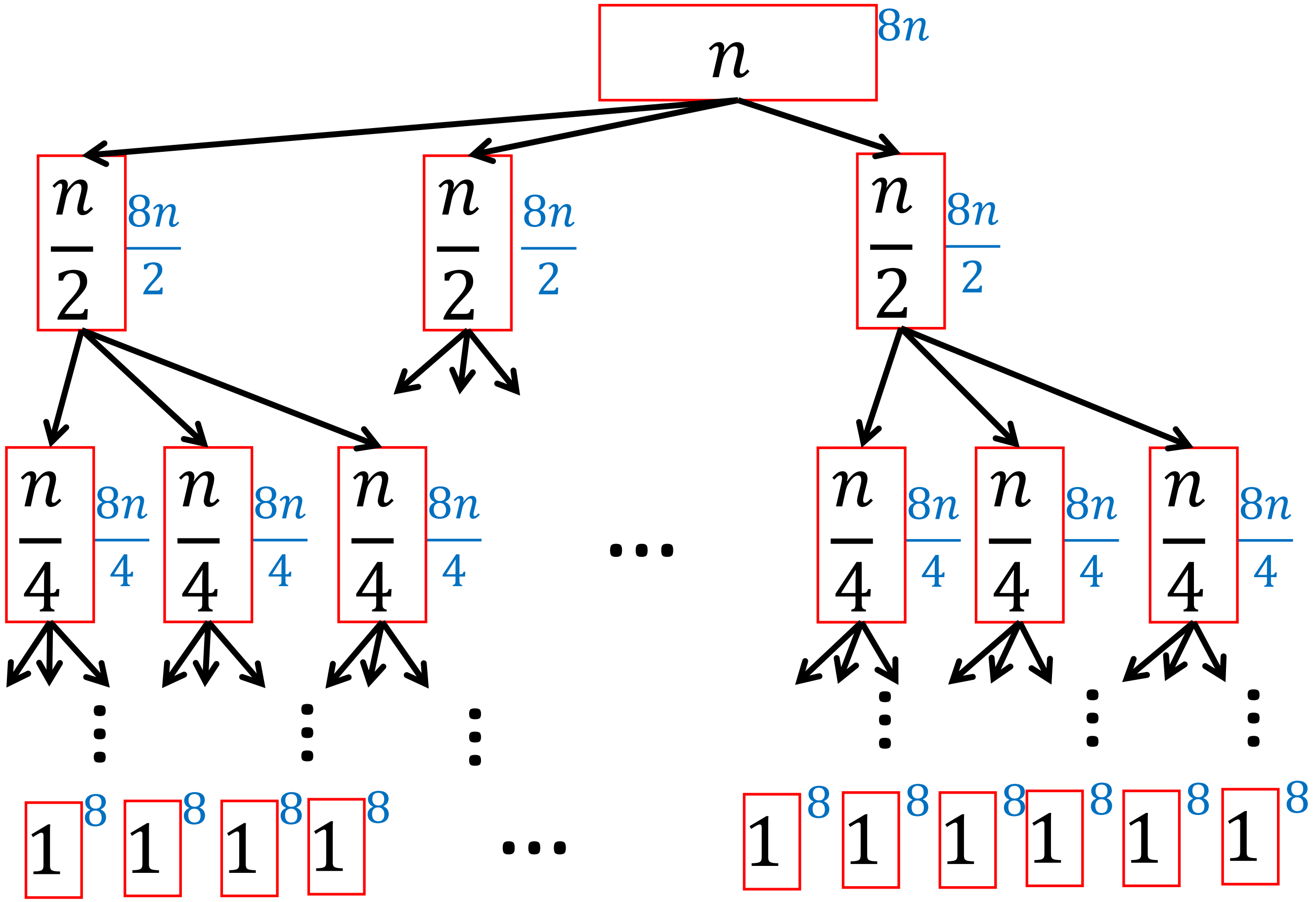
1. x = Karatsuba(a,c)
2. y = Karatsuba(a,d)
3. z = Karatsuba(a+b,c+d)-x-y
4. Return 10^nx +$10^{n/2}$z+y

Karatsuba

$$T(n) = 8n \sum_{i=0}^{\log_2 n} \left({}^{3}/_{2}\right)^{i}$$

3. Use asymptotic notation to simplify

$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$



$$8 \cdot 1n$$

$$\frac{8}{2} \cdot 3n$$

$$\frac{8}{4} \cdot 9n$$

$$\vdots$$

$$\frac{8}{2^{\log_2 n}} \cdot 3^{\log_2 n} n$$

Karatsuba

3. Use asymptotic notation to simplify

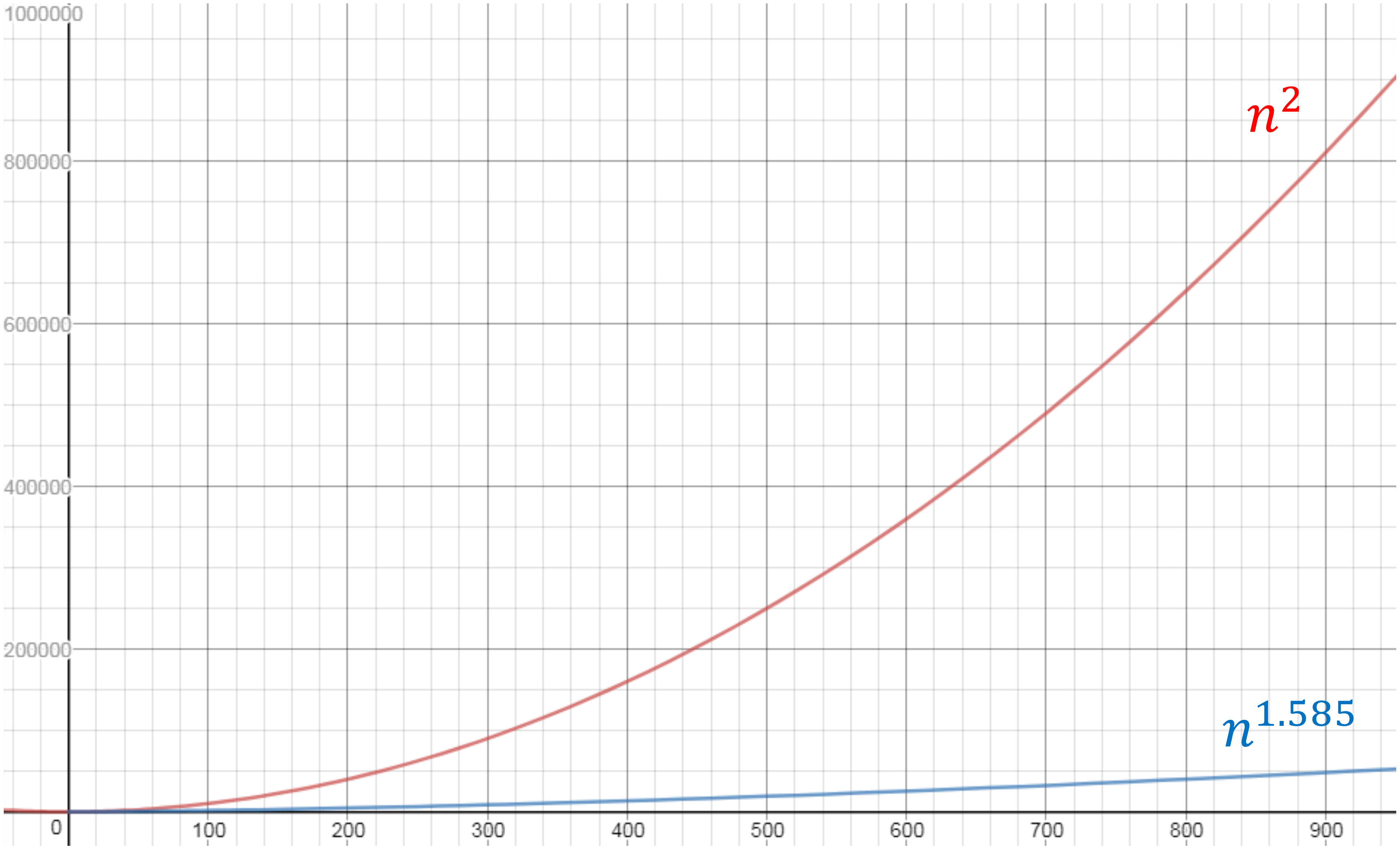
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

$$T(n) = 8n \sum_{i=0}^{\log_2 n} (^3/_2)^i$$

$$T(n) = 8n \frac{(^3/_2)^{\log_2 n+1} - 1}{^3/_2 - 1}$$

Math, math, and more math…(on board, see lecture supplemental)

$$T(n) = 24\left(n^{\log_2 3}\right) - 16n \;= \Theta(n^{\log_2 3}) \approx \Theta(n^{1.585})$$



1000000
800000
600000
400000
200000
0
100
200
300
400
500
600
700
800
900
n^2
$n^{1.585}$

Recurrence Solving Techniques

Tree

?✓ Guess/Check (induction)

“Cookbook”

Substitution

Induction (review)

Goal: $\forall k, P(k)$ holds

Base case(s): $P(1)$ holds

Hypothesis: $\forall x \leq x_0, P(x)$ holds

Inductive step: $P(x_0) \Rightarrow P(x_0 + 1)$

Guess and Check Intuition

- To Prove: $T(n) = O(g(n))$
- Consider: $g_*(n) = O(g(n))$
- Goal: show $\exists n_0$ s.t. $\forall n > n_0, T(n) \leq g_*(n)$
  - (definition of big-O)
- Technique: Induction
  - Base cases:
    - show $T(1) \leq g_*(1), T(2) \leq g_*(2)$, … for a small number of cases
  - Hypothesis:
    - $\forall n \leq x_0, T(n) \leq g_*(n)$
  - Inductive step:
    - $T(x_0 + 1) \leq g_*(x_0 + 1)$

Karatsuba Guess and Check (Loose)

$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Goal: $T(n) \leq 3000\, n^{1.6} = O(n^{1.6})$

Base cases: $T(1) = 8 \leq 3000$
$T(2) = 3(8) + 16 = 40 \leq 3000 \cdot 2^{1.6}$
… up to some small k

Hypothesis: $\forall n \leq x_0, T(n) \leq 3000n^{1.6}$

Inductive step: $T(x_0 + 1) \leq 3000(x_0 + 1)^{1.6}$

Math, math, and more math…(on board, see lecture supplemental)

Karatsuba Guess and Check (Loose)

Goal $T(x_0+1) \le 3000\,(x_0+1)^{1.6}$ $\qquad$ $T(n) = 3T\left(\frac{n}{2}\right) + 8n$

$\frac{x_0+1}{2} < x_0$

$$T(x_0+1) = 3T\left(\frac{x_0+1}{2}\right) + 8(x_0+1)$$

$$\le 3\cdot\left(3000\cdot\left(\frac{x_0+1}{2}\right)^{1.6}\right) + 8(x_0+1)$$

$$= \frac{3}{2^{1.6}}\left[3000\,(x_0+1)^{1.6}\right] + 8\,(x_0+1)$$

$0.997 = 1 - .003$

$$\le 0.997\left(3000\,(x_0+1)^{1.6}\right) + 8\,(x_0+1)$$

$$= (1-.003)\left(3000(x_0+1)^{1.6}\right) + 8\,(x_0+1)$$

$.003 \times 3000 = 9$

$$= 3000(x_0+1)^{1.6} + 8(x_0+1) - 9(x_0+1)^{1.6}$$

$$\le 3000(x_0+1)^{1.6}$$

minus something

$8 < 9$

$(x_0+1) < (x_0+1)^{1.6}$

Karatsuba Guess and Check (Loose)

$$T(n) \leq 3000(n)^{1.6}$$

$$T(n) = O(n^{1.6})$$

Mergesort Guess and Check

$$T(n) = 2T(\frac{n}{2}) + n$$

Goal: $T(n) \le n \log_2 n \ = O(n \log_2 n)$

Base cases: $T(1) = 0$
$T(2) = 2 \le 2 \log_2 2$
… up to some small k

Hypothesis: $\forall n \le x_0 \; T(n) \le n \log_2 n$

Inductive step: $T(x_0 + 1) \le (x_0 + 1) \log_2(x_0 + 1)$

Math, math, and more math…(on board, see lecture supplemental)

Mergesort Guess and Check

$\forall n < x_0$

$T(n) \le n \log n$

$$T(x_0+1) = 2T\left(\frac{x_0+1}{2}\right) + (x_0+1)$$

$\frac{x_0+1}{2} < x_0$

$$\le 2 \cdot \left(\frac{x_0+1}{2}\right) \log_2\left(\frac{x_0+1}{2}\right) + (x_0+1)$$

$$= (x_0+1) \log_2\left(\frac{x_0+1}{2}\right) + (x_0+1)$$

$$= (x_0+1)\left(\log_2(x_0+1) - \log_2 2\right) + (x_0+1)$$

$\log_2 2 = 1$

$$= (x_0+1)\left(\log_2(x_0+1)\right) - (x_0+1) + (x_0+1)$$

$$= (x_0+1) \log_2(x_0+1)$$

$$T(x_0+1) \le (x_0+1) \log_2(x_0+1)$$

Karatsuba Guess and Check

$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Goal: $T(n) \leq 24n^{\log_2 3} - 16n = O(n^{\log_2 3})$

Base cases: by inspection, holds for small n (at home)

Hypothesis: $\forall n \leq x_0, T(n) \leq 24n^{\log_2 3} - 16n$

Inductive step: $T(x_0 + 1) \leq 24(x_0 + 1)^{\log_2 3} - 16(x_0 + 1)$

Math, math, and more math…(on board, see lecture supplemental)

Karatsuba Guess and Check

hyp: $T(n) \le 24 n^{\log_2 3} - 16n$
$\forall n \le x_0$

$T(n) = 3T\left(\frac{n}{2}\right) + 8n$
$= O(n^{\log_2 3})$

$2^{\log_2 3} = 3$

$$T(x_0+1) = 3 \cdot T\left(\frac{x_0+1}{2}\right) + 8(x_0+1)$$
$$\le 3 \cdot \left(24\left(\frac{x_0+1}{2}\right)^{\log_2 3} - 16\left(\frac{x_0+1}{2}\right)\right) + 8(x_0+1)$$
$$= 3\left(\frac{24}{3}(x_0+1)^{\log_2 3}\right) - 24(x_0+1) + 8(x_0+1)$$
$$= 24(x_0+1)^{\log_2 3} - 16(x_0+1)$$

What if we leave out the $-16n$?

$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Goal: $T(n) \leq 24n^{\log_2 3} - 16n = O(n^{\log_2 3})$

Base cases: by inspection, holds for small n (at home)

Hypothesis: $\forall n \leq x_0, T(n) \leq 24n^{\log_2 3} - 16n$

Inductive step: $T(x_0 + 1) \leq 24(x_0 + 1)^{\log_2 3} - 16(x_0 + 1)$

What we wanted: $T(x_0 + 1) \leq 24(x_0 + 1)^{\log_2 3}$ Induction failed!

What we got: $T(x_0 + 1) \leq 24(x_0 + 1)^{\log_2 3} + 8(x_0 + 1)$

"Bad Mergesort" Guess and Check

$$T(n) = 2T(\frac{n}{2}) + 209n$$

Goal: $T(n) \le 209n \log_2 n \; = O(n \log_2 n)$

Base cases: $T(1) = 0$
$T(2) = 518 \le 209 \cdot 2 \log_2 2$
… up to some small k

Hypothesis: $\forall n \le x_0, T(n) \le 209n \log_2 n$

Inductive step: $T(x_0 + 1) \le 209(x_0 + 1) \log_2(x_0 + 1)$

Recurrence Solving Techniques

Tree

Guess/Check

"Cookbook"

Substitution

Observation

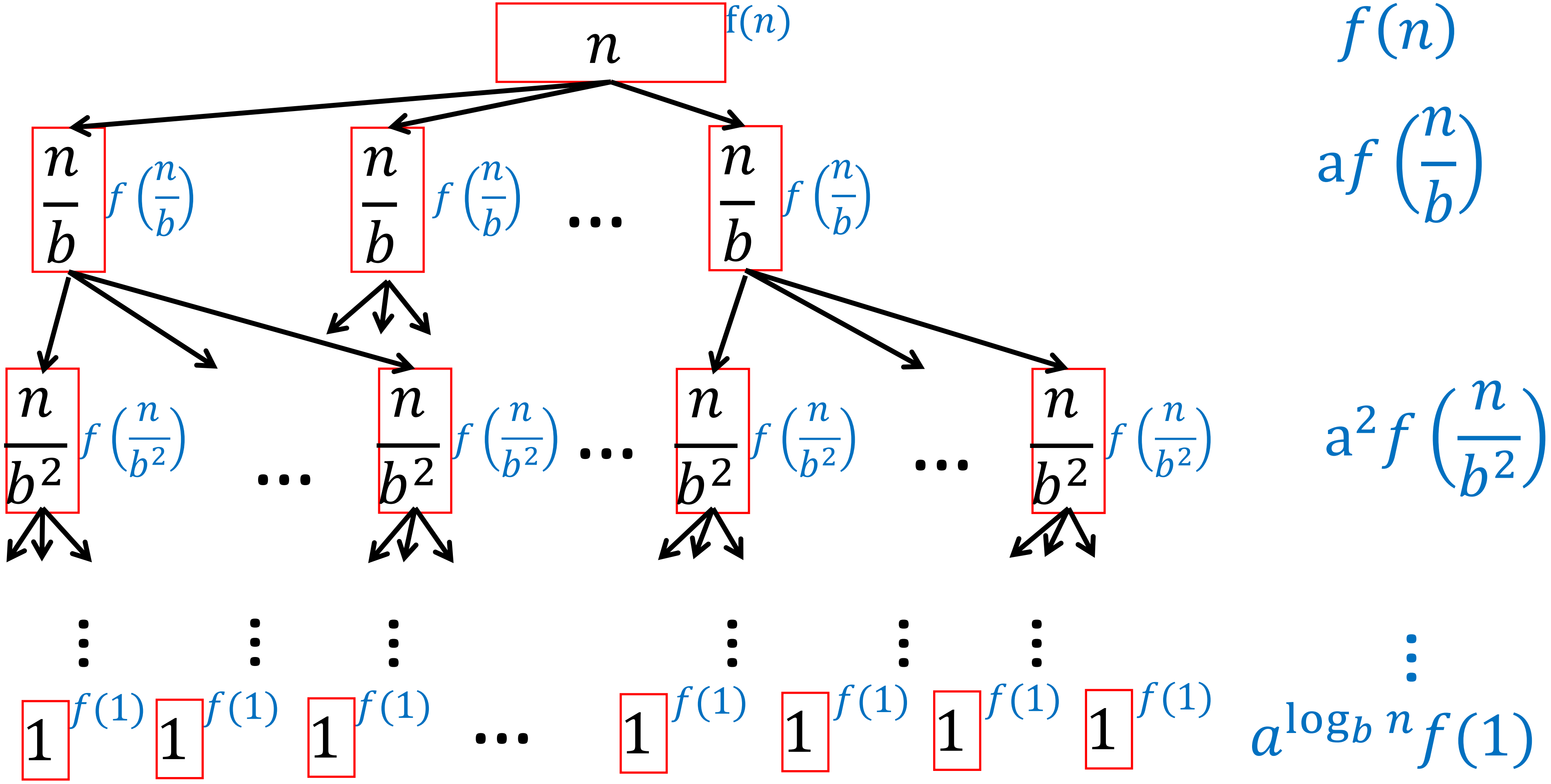
- **Divide:** $D(n)$ time,
- **Conquer:** recurse on small problems, size s
- **Combine:** $\mathrm{C}(n)$ time
- **Recurrence:**
  - $T(n) = D(n) + \sum T(s) + C(n)$

- Many D&C recurrences are of form:
  - $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

General

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$$T(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$



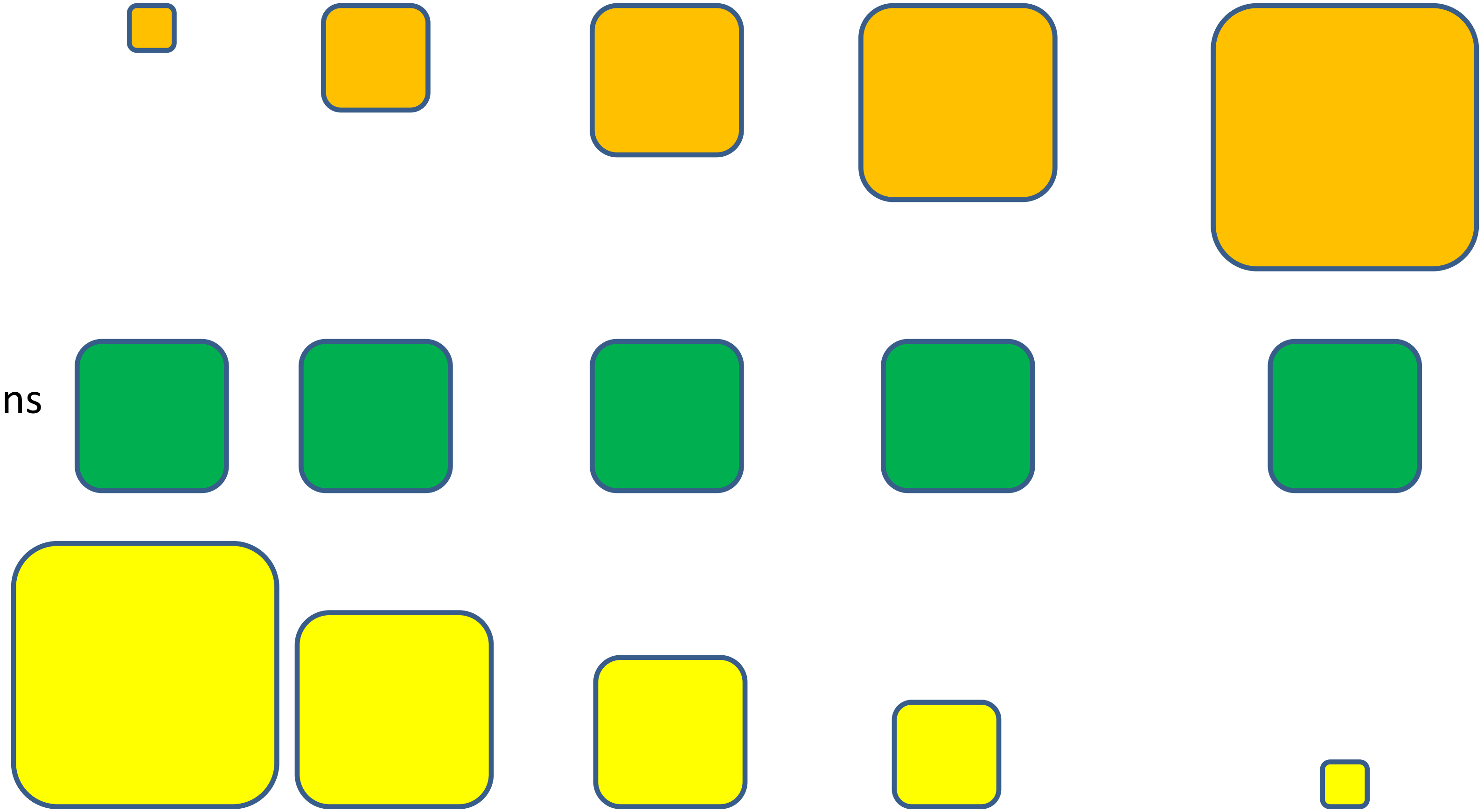
3 Cases

$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + a^3 f\left(\frac{n}{b^3}\right) + \cdots + a^L f\left(\frac{n}{b^L}\right)$$

Case 1:
Most work happens at the leaves

Case 2:
Work happens consistently throughout

Case 3:
Most work happens at top of tree



Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- Case 2: if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$