CS4102 Algorithms

Spring 2022

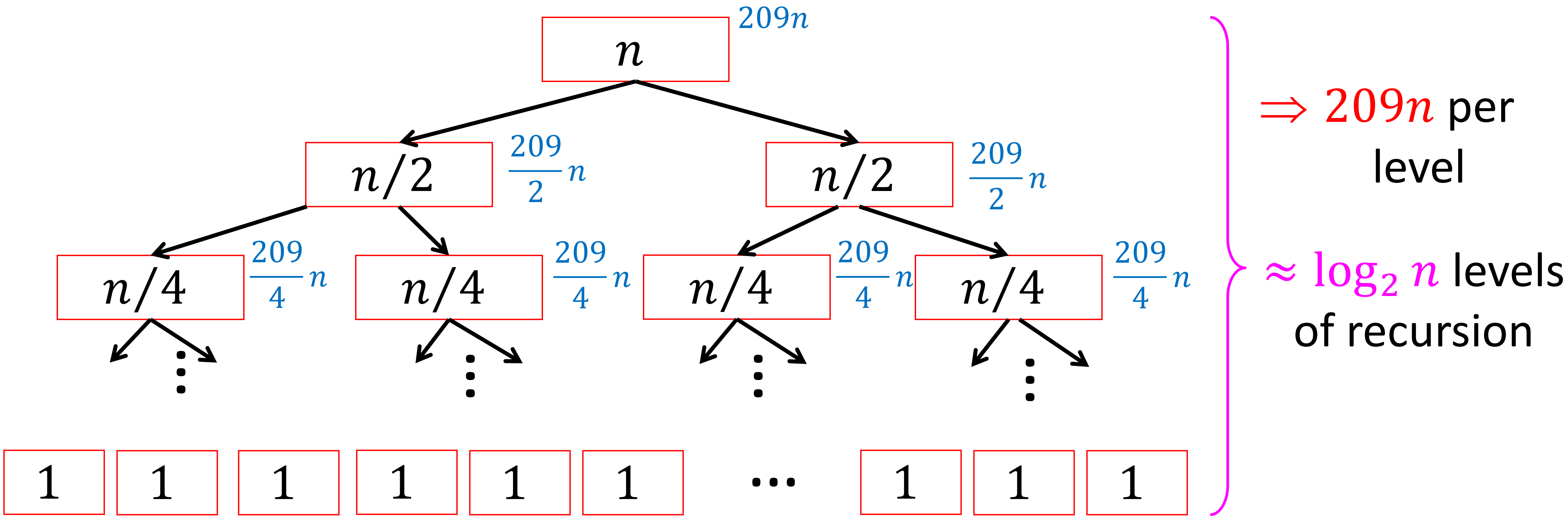
Warm Up

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

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

Tree Method

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



$$T(n) = 209n \sum_{i=0}^{\log n} 1 = 209n \log_2 n$$

Tree Method

$$T(n) = 2T(n/2) + 209n$$

What is the cost?

Cost at level i: $2^i \cdot \frac{209n}{2^i} = 209n$

Total cost: $T(n) = \sum_{i=0}^{\log_2 n} 209n$

$$= 209n \sum_{i=0}^{\log_2 n} 1 = n \log_2 n$$

$$= \Theta(n \log n)$$

Number of subproblems	Cost of subproblem
1	$209n$
2	$209n/2$
4	$209n/4$
2^k	$209n/2^k$

Multiplication

- Want to multiply large numbers together

$$\begin{array}{r} 4\,1\,0\,2 \\ \times\,1\,8\,1\,9 \\ \hline \end{array}$$

n-digit numbers

- What makes a "good" algorithm?
- How do we measure input size?
- What do we "count" for run time?

"Schoolbook" Method

Can we do better?

How many total multiplications?

$$
\begin{array}{r}
4\,1\,0\,2 \\
\times\ 1\,8\,1\,9 \\
\hline
3\,6\,9\,1\,8 \\
4\,1\,0\,2 \\
3\,2\,8\,1\,6 \\
+\ 4\,1\,0\,2 \\
\hline
7\,4\,6\,1\,5\,3\,8
\end{array}
$$

n-digit numbers

n mults

n mults

n mults

n mults

n levels $\Rightarrow \Theta(n^2)$

What about cost of additions?

$\Theta(n^2)$

Divide and Conquer method

1. Break into smaller subproblems

$$\begin{array}{r} ab \\ \times\ cd \\ \hline \end{array} \qquad \begin{aligned} ab &= 10^{\frac{n}{2}}\, a + b \\ cd &= 10^{\frac{n}{2}}\, c + d \end{aligned}$$

$$10^{n}(a \times c) +$$

$$10^{\frac{n}{2}}(a \times d + b \times c) +$$

$$(b \times d)$$

Divide and Conquer Multiplication

- **Divide**:
  - Break n-digit numbers into four numbers of $^n/_2$ digits each (call them a, b, c, d)
- **Conquer**:
  - If $n > 1$:
    - Recursively compute ac, ad, bc, bd
  - If $n = 1$: (i.e. one digit each)
    - Compute ac, ad, bc, bd directly (base case)
- **Combine**:

$$10^n(ac) + 10^{\frac{n}{2}}(ad + bc) + bd$$

Divide and Conquer method

2. Use recurrence relation to express recursive running time

$$10^n(\boxed{ac}) + 10^{\frac{n}{2}}(\boxed{ad} + \boxed{bc}) + \boxed{bd}$$

Recursively solve

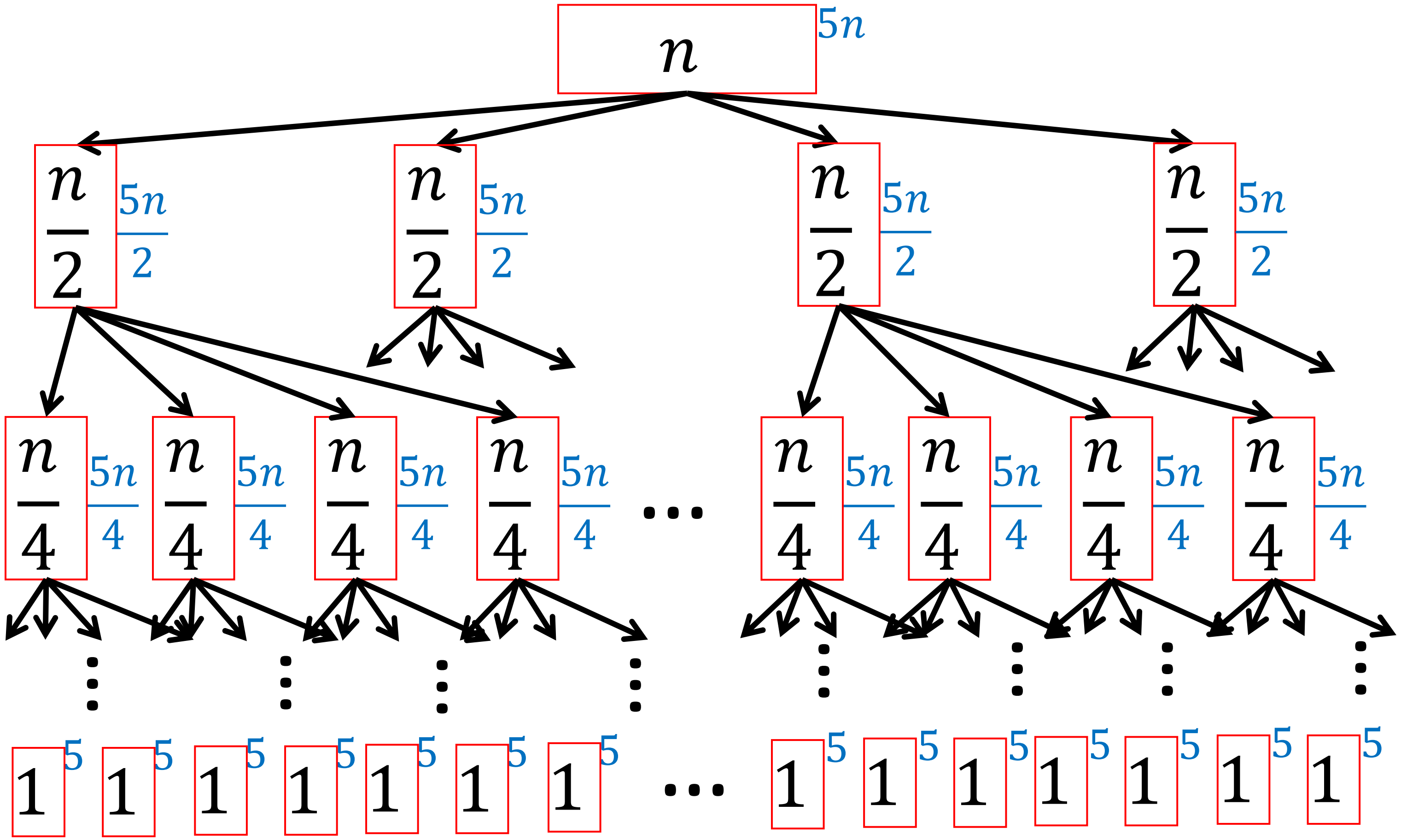
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Divide and Conquer method

3. Use asymptotic notation to simplify

$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

$$T(n) = 5n \sum_{i=0}^{\log_2 n} 2^i$$



$$5n$$

$$\frac{4}{2} \cdot 5n$$

$$\frac{16}{4} \cdot 5n$$

$$\vdots$$

$$2^{\log_2 n} \cdot 5n$$

Divide and Conquer method

3. Use asymptotic notation to simplify

$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

$$T(n) = 5n \sum_{i=0}^{\log_2 n} 2^i$$

$$T(n) = 5n \frac{2^{\log_2 n+1} - 1}{2 - 1}$$

$$T(n) = 5n(2n - 1) \ = \Theta(n^2)$$

Karatsuba Multiplication

1. Break into smaller subprojects

$$\begin{array}{r} ab \\ \times\ cd \\ \hline \end{array} \qquad \begin{aligned} ab &= 10^{\frac{n}{2}} a + b \\ cd &= 10^{\frac{n}{2}} c + d \end{aligned}$$

$$10^{n}(a \times c) +$$

$$10^{\frac{n}{2}}(a \times d + b \times c) +$$

$$(b \times d)$$

Karatsuba

$$\begin{array}{r} a \quad b \\ \times \quad c \quad d \\ \hline \end{array}$$

$$10^n(ac) + 10^{\frac{n}{2}}(ad + bc) + bd$$

Can't avoid these

This can be simplified

$$(a + b)(c + d) =$$

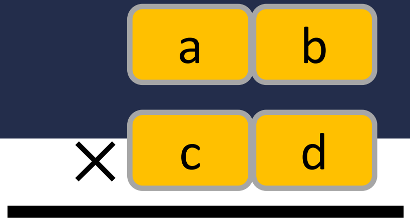
$$ac + ad + bc + bd$$

$$ad + bc = (a + b)(c + d) - ac - bd$$

Two multiplications

One multiplication

Karatsuba Algorithm



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

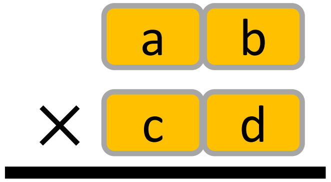
Pseudo-code

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

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

Karatsuba Multiplication

2. Use recurrence relation to express recursive running time



$$10^{n}(ac) + 10^{n/2}\big((a+b)(c+d) - ac - bd\big) + bd$$

Recursively solve

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

Need to compute **3** multiplications, each of size $n/2$: ac, bd, $(a+b)(c+d)$

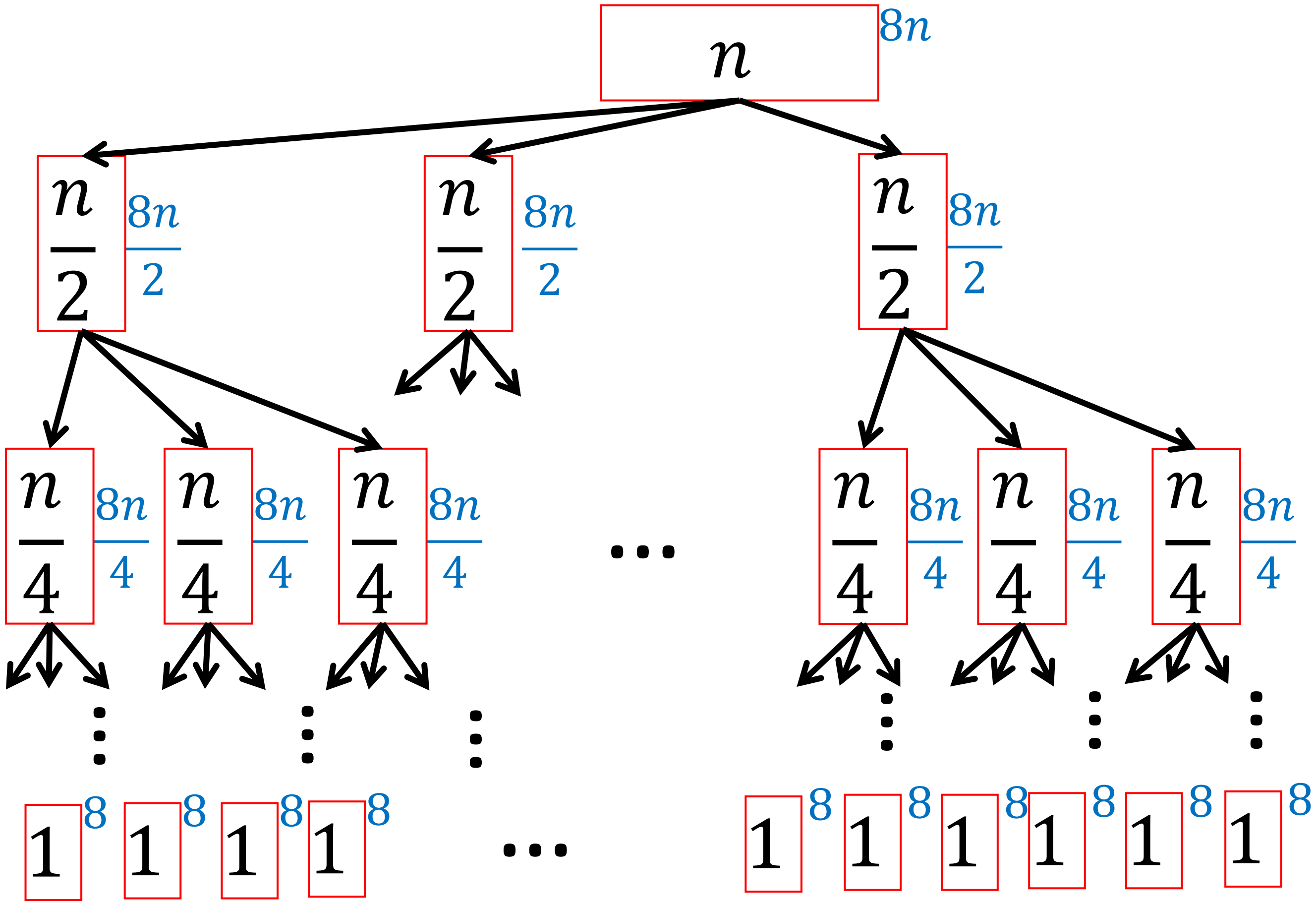
2 shifts and 6 additions on n-digit values

Karatsuba

3. Use asymptotic notation to simplify

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

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



$$8n \cdot 1$$

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

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

$$\vdots$$

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

Karatsuba

3. Use asymptotic notation to simplify

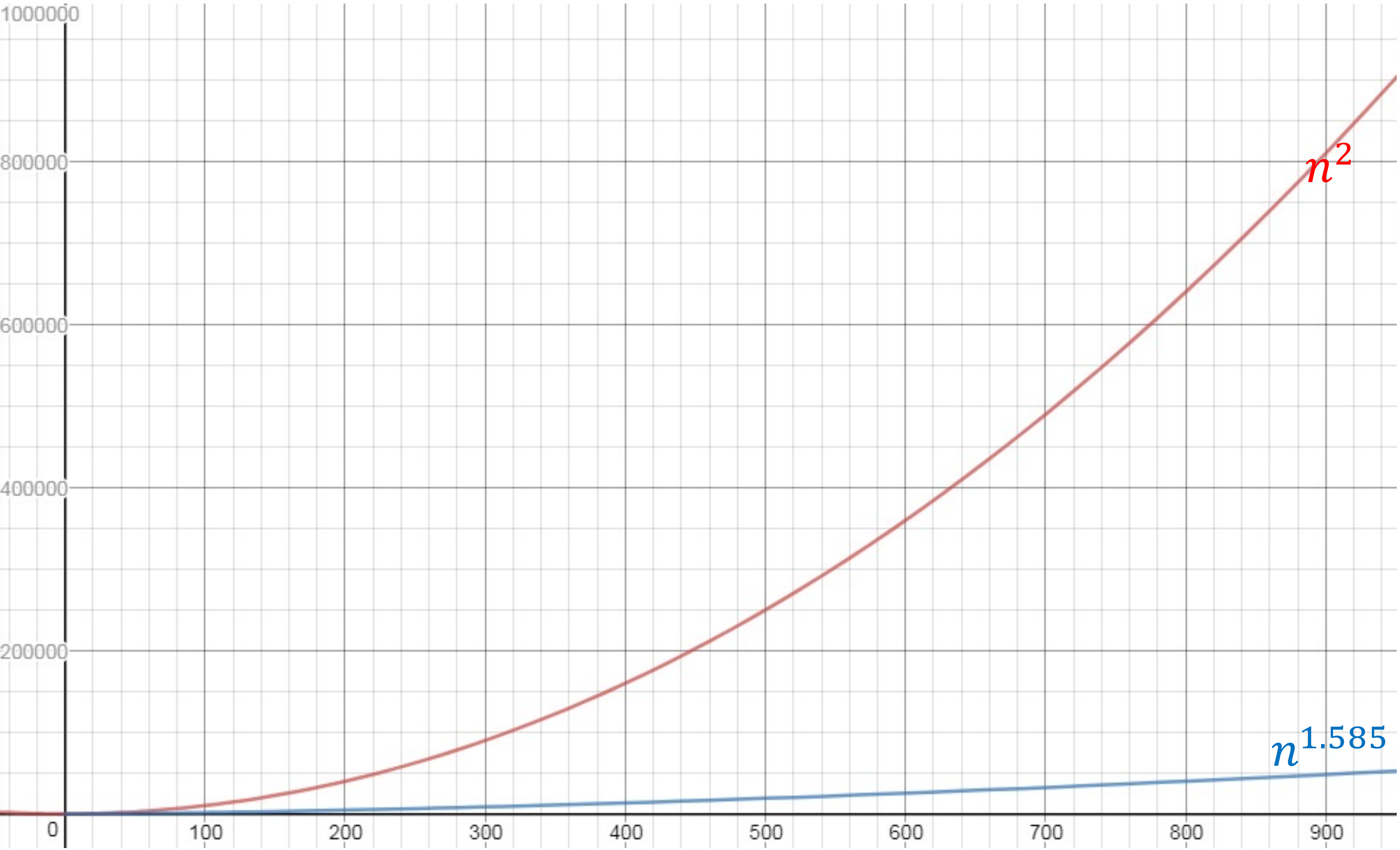
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

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

$$T(n) = 8n \frac{\left({}^{3}/_{2}\right)^{\log_2 n+1} - 1}{{}^{3}/_{2} - 1}$$

Math, math, and more math…(see lecture supplement)

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



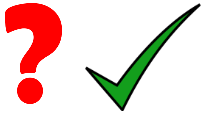
1000000
800000
600000
400000
200000
0
100
200
300
400
500
600
700
800
900
n^2
$n^{1.585}$

Recurrence Solving Techniques

Four methods for solving recurrences

- Unrolling: expand the recurrence

- Tree: get a picture of recursion



- Guess/Check: Substitution by guessing the solution and using induction to prove

- “Cookbook”: Use magic (a.k.a. Master Theorem)

Induction (review)

Goal: $\forall k \in \mathbb{N}, P(k)$ holds

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

Technically, called *strong induction*

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

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

Guess and Check Intuition

- **Show:** $T(n) \in O(g(n))$
- **Consider:** $g_*(n) = c \cdot g(n)$ for some constant c, i.e. pick $g_*(n) \in O(g(n))$
- **Goal:** show $\exists n_0$ such that $\forall n > n_0, T(n) \leq g_*(n)$
  - (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 (may need additional base cases)
  - Hypothesis:
    - $\forall n \leq x_0, T(n) \leq g_*(n)$
  - Inductive step:
    - Show $T(x_0 + 1) \leq g_*(x_0 + 1)$

Need to ensure that in inductive step, can either appeal to a <u>base case</u> or to the <u>inductive hypothesis</u>

Karatsuba Guess and Check (Loose)

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

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: Show that $T(x_0 + 1) \leq 3000(x_0 + 1)^{1.6}$

Karatsuba Guess and Check (Loose)

Mergesort Guess and Check

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

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

Base cases: $T(1) = 0$

$T(2) = 2 \leq 2 \log_2 2$

… up to some small k

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

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

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

Mergesort Guess and Check

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

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!

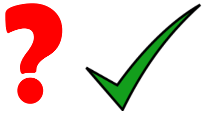
What we got: $T(x_0 + 1) \leq 24(x_0 + 1)^{\log_2 3} + 8(x_0 + 1)$

Recurrence Solving Techniques

Four methods for solving recurrences

- Unrolling: expand the recurrence

- Tree: get a picture of recursion



- Guess/Check: Substitution by guessing the solution and using induction to prove

- “Cookbook”: Use magic (a.k.a. Master Theorem)

Observation

- **Divide**: $D(n)$ time
- **Conquer**: recurse on small problems, size s
- **Combine**: $C(n)$ time
- **Recurrence:**

$$T(n) = D(n) + \sum T(s) + C(n)$$

- Many D&C recurrences are of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \qquad \text{where } f(n) = D(n) + C(n)$$

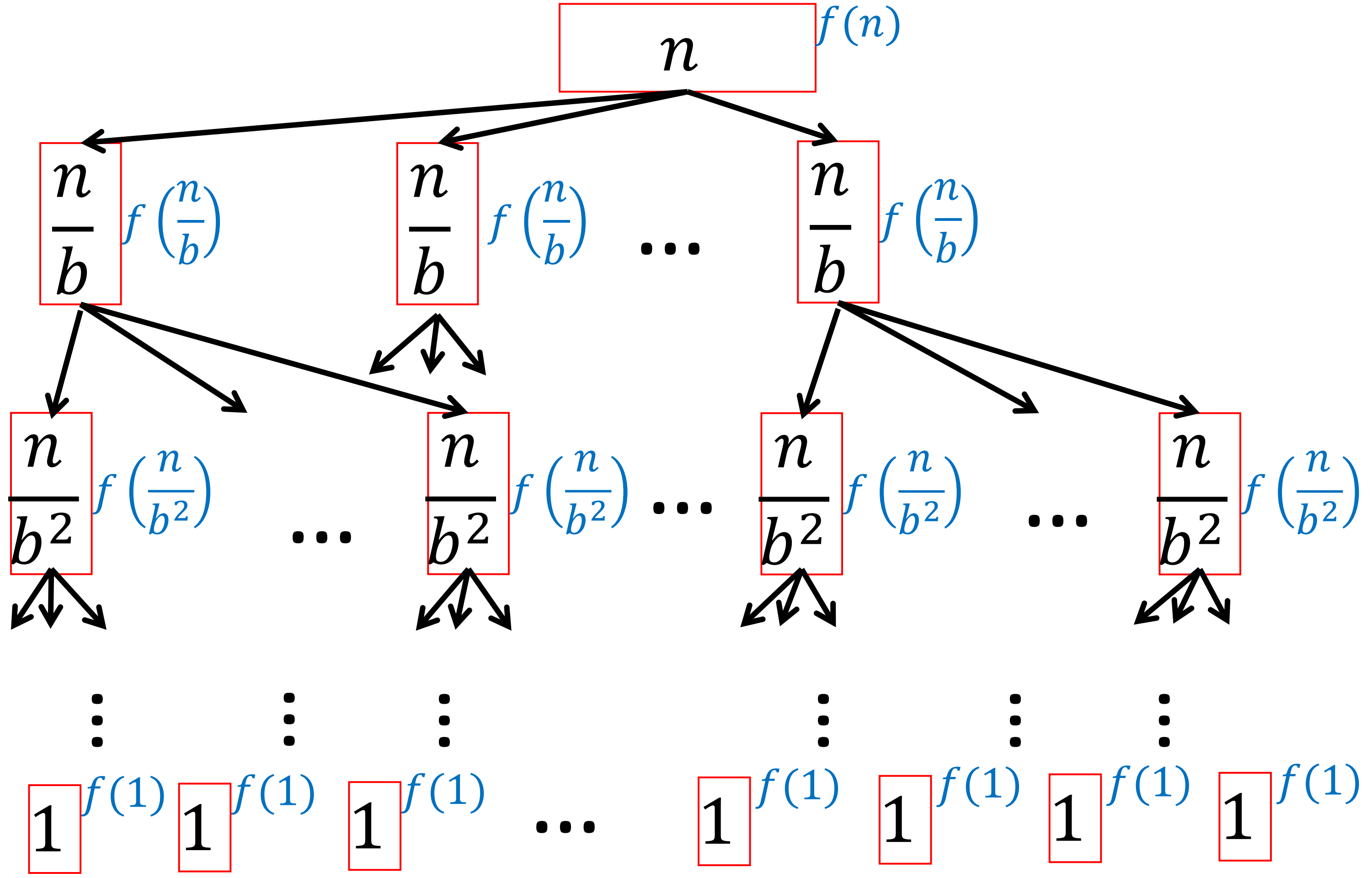
Remember…

- MergeSort: $T(n) = 2\,T\left(\frac{n}{2}\right) + n$
- D&C Multiplication: $T(n) = 4T\left(\frac{n}{2}\right) + 5n$
- Karatsuba: $T(n) = 3T\left(\frac{n}{2}\right) + 8n$

General

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

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



$$f(n)$$

$$af\left(\frac{n}{b}\right)$$

$$a^2 f\left(\frac{n}{b^2}\right)$$

$$\vdots$$

$$a^{\log_b n} f(1)$$

3 Cases

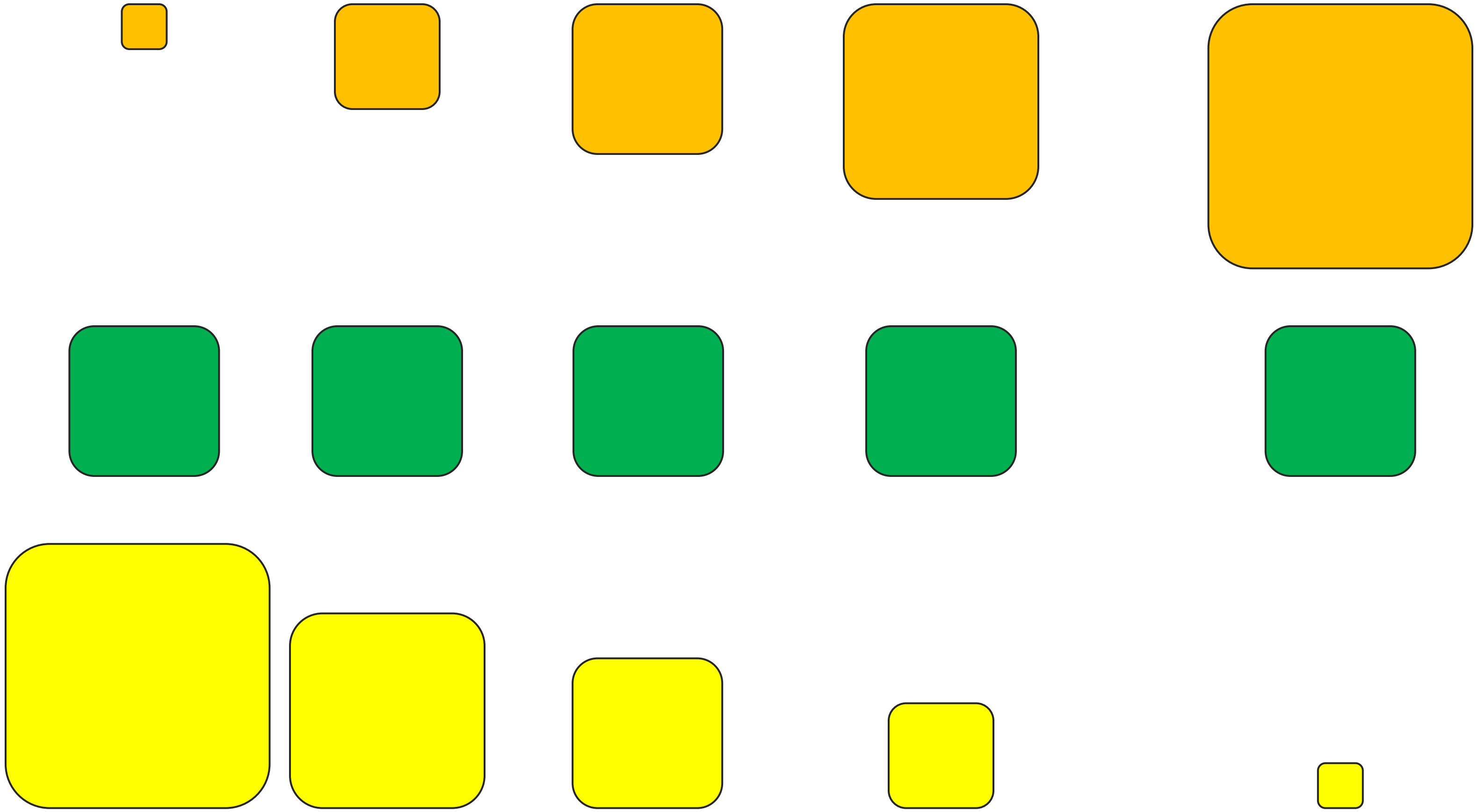
$$L = \log_b n$$

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

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 $f(n) \in O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$,
then $T(n) \in \Theta\left(n^{\log_b a}\right)$

Case 2: if $f(n) \in \Theta(n^{\log_b a})$, then $T(n) \in \Theta(n^{\log_b a} \log n)$

Case 3: if $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ 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 $T(n) \in \Theta(f(n))$

Master Theorem Example 1

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

- 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 $T(n) = \Theta(f(n))$

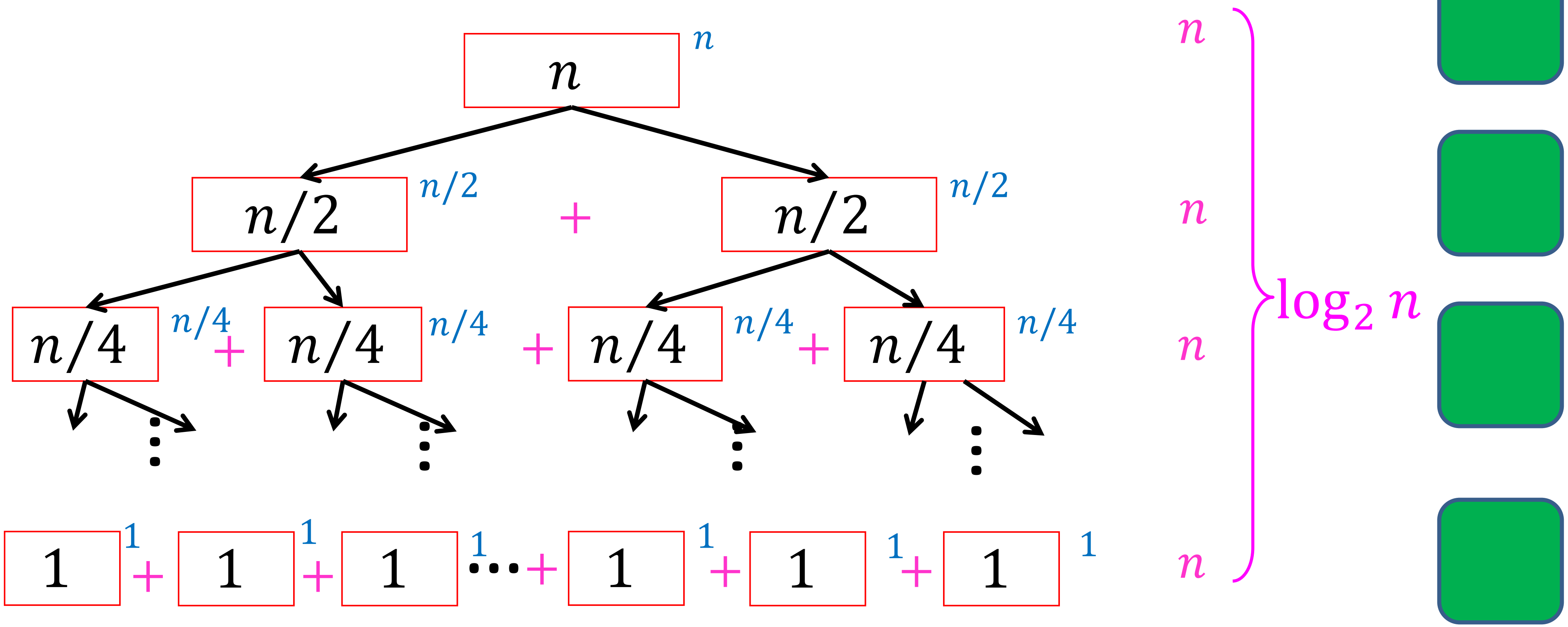
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Case 2

$$\Theta\left(n^{\log_2 2} \log n\right) = \Theta(n \log n)$$

Tree method

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



Master Theorem Example 2

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

- 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 $T(n) = \Theta(f(n))$

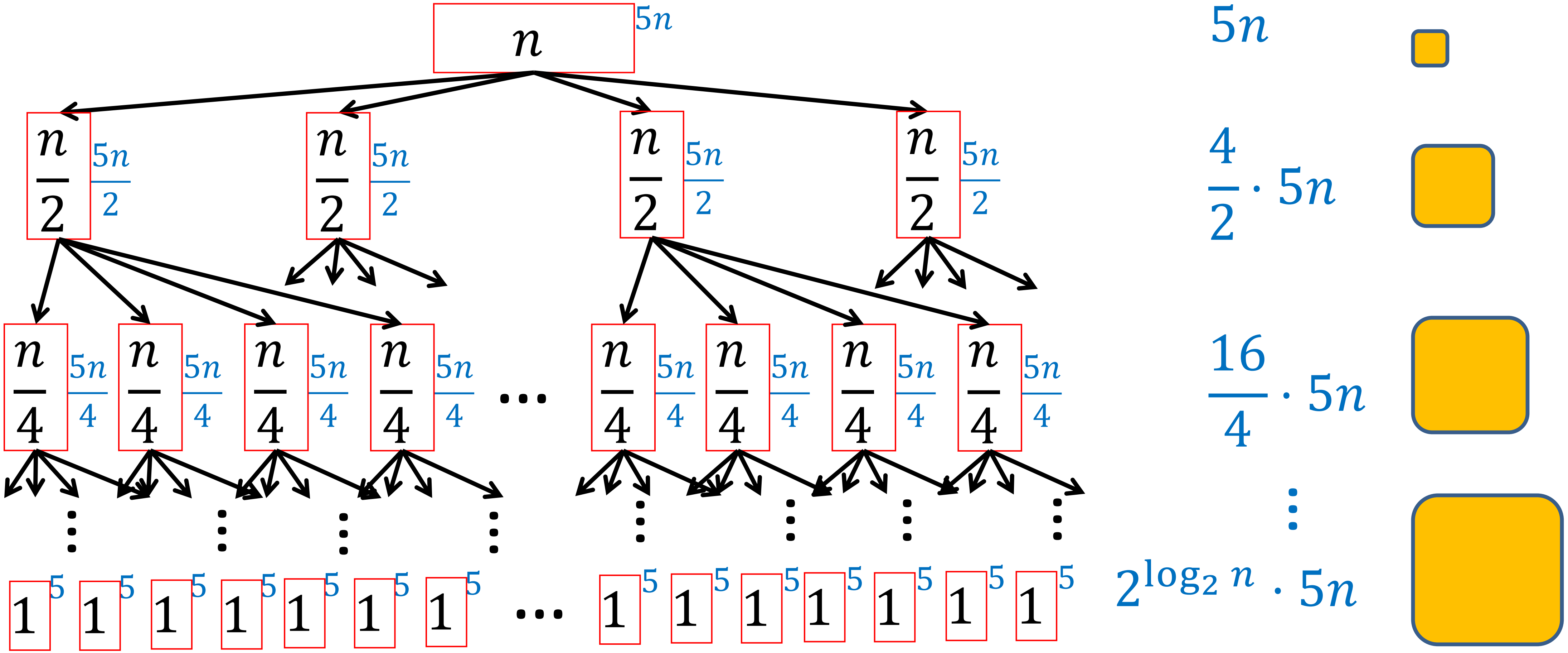
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Case 1

$$\Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

Tree method

$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$



Master Theorem Example 3

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

- 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 $T(n) = \Theta(f(n))$

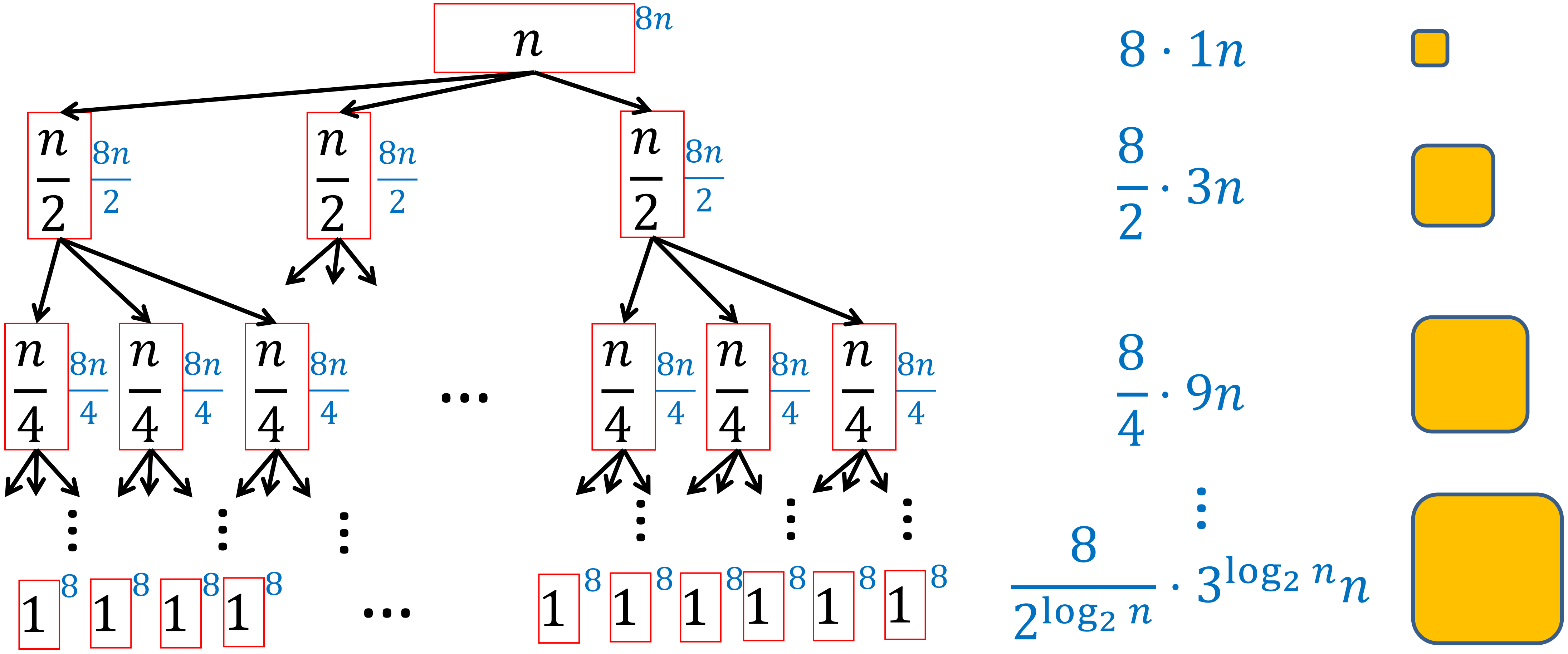
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Case 1

$$\Theta\left(n^{\log_2 3}\right) \approx \Theta(n^{1.5})$$

Karatsuba

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



Master Theorem Example 4

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

- 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 $T(n) = \Theta(f(n))$

$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$

Case 3

$$\Theta(n^3)$$

Tree method

$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$