CS4102 Algorithms

Fall 2018

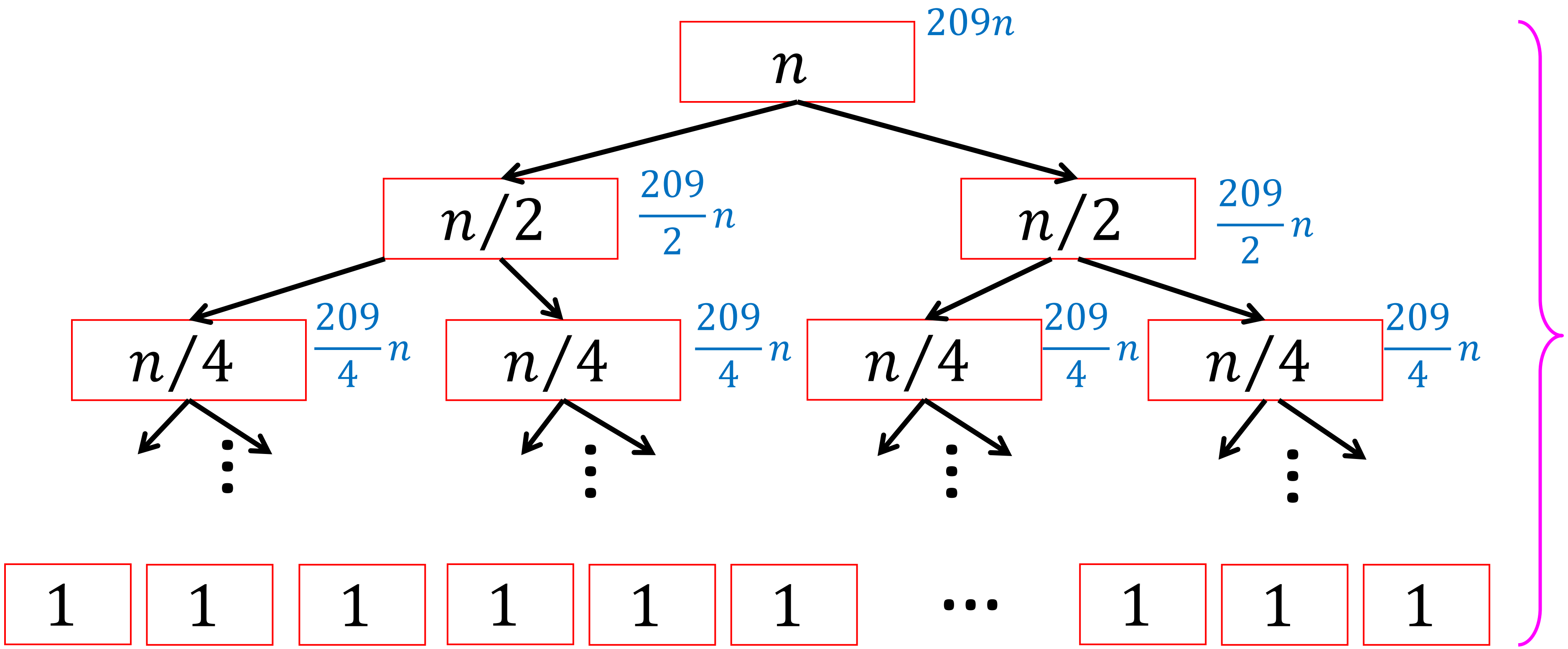
Warm Up

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

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

Tree method

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



$\Rightarrow$ $209n$ per level

$\log_2 n$ levels of recursion

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

Today’s Keywords

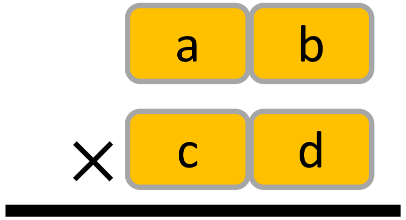
- Karatsuba
- Guess and check Method
- Induction
- Master Theorem

CLRS Readings

- Chapter 4

Homework

- Hw1 due 11pm Wednesday, Sept 12
  - Written (use Latex!)
  - 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 $10^n(ac) + 10^{\frac{n}{2}}(ad+bc) + bd$

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

<u>Pseudo-code</u>

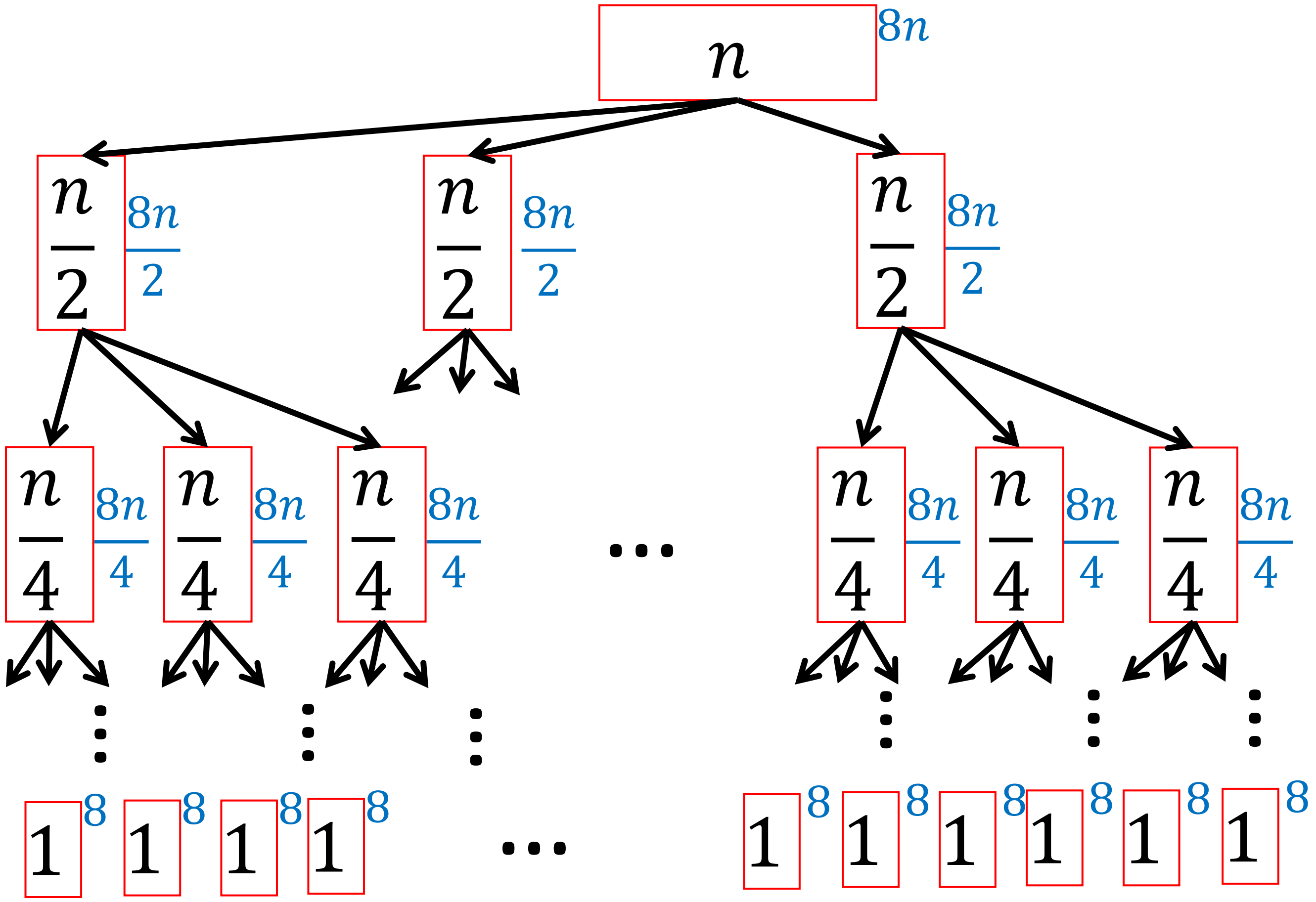
1. x = Karatsuba(a,c)
2. y = Karatsuba(a,d)
3. z = Karatsuba(a+b,c+d)-x-y
4. Return 10^nx +$10^{n/2}$z+y

Karatsuba

3. Use asymptotic notation to simplify

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

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



$$8 \cdot 1n$$

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

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

$$\vdots$$

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

Karatsuba

3. Use asymptotic notation to simplify

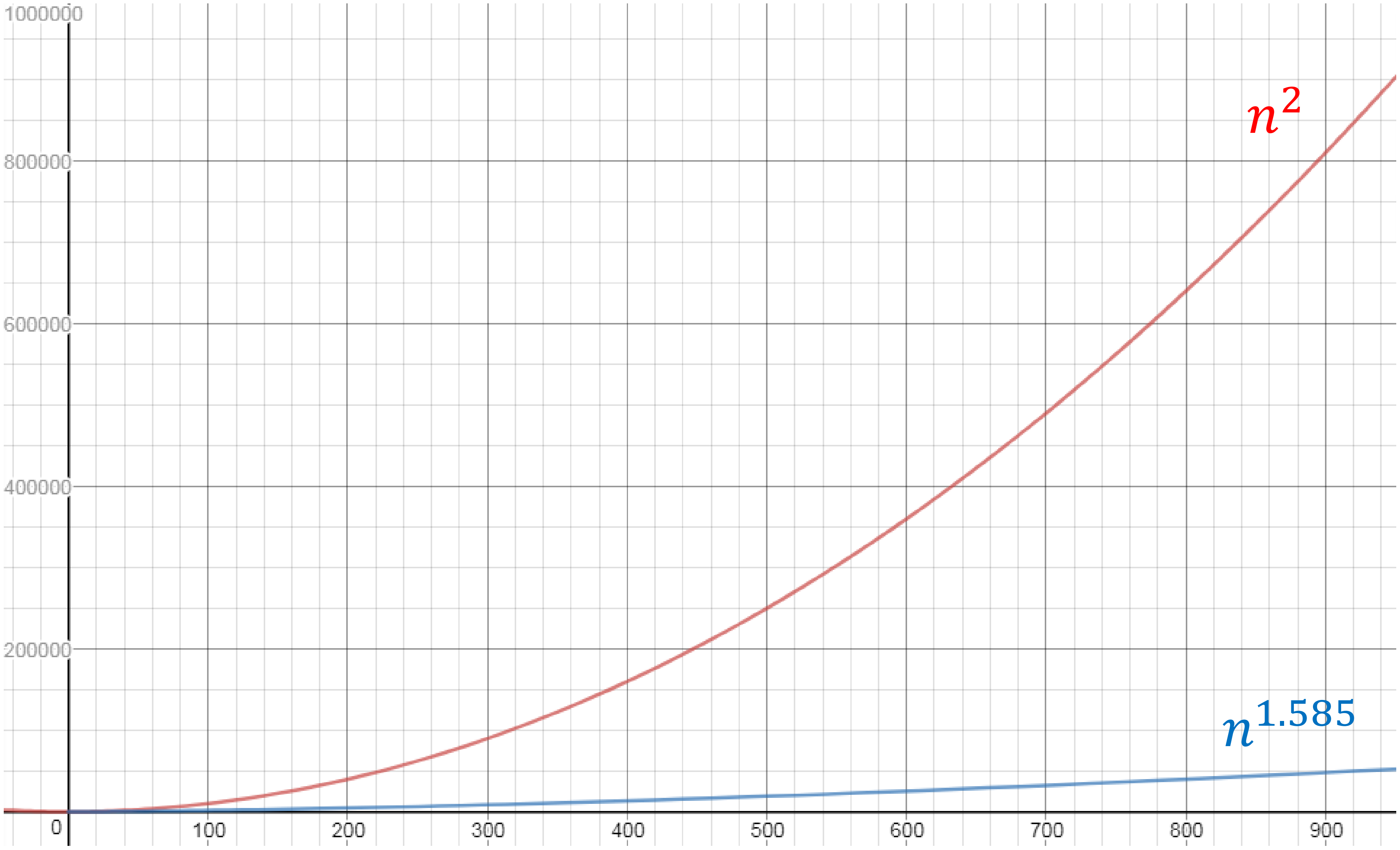
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

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

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

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

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


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

Recurrence Solving Techniques


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. $\forall n > n_0, T(n) \le g_*(n)$
  - (definition of big-O)
- Technique: Induction
  - Base cases:
    - show $T(1) \le g_*(1), T(2) \le g_*(2)$, … for a small number of cases
  - Hypothesis:
    - $\forall n \le x_0, T(n) \le g_*(n)$
  - Inductive step:
    - $T(x_0 + 1) \le g_*(x_0 + 1)$

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

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

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

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: $T(n) \leq 209n \log_2 n \ = O(n \log_2 n)$

Base cases: $T(1) = 0$

$T(2) = 518 \leq 209 \cdot 2 \log_2 2$

… up to some small k

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

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

Recurrence Solving Techniques


Tree


Guess/Check

"Cookbook"

Substitution

Observation

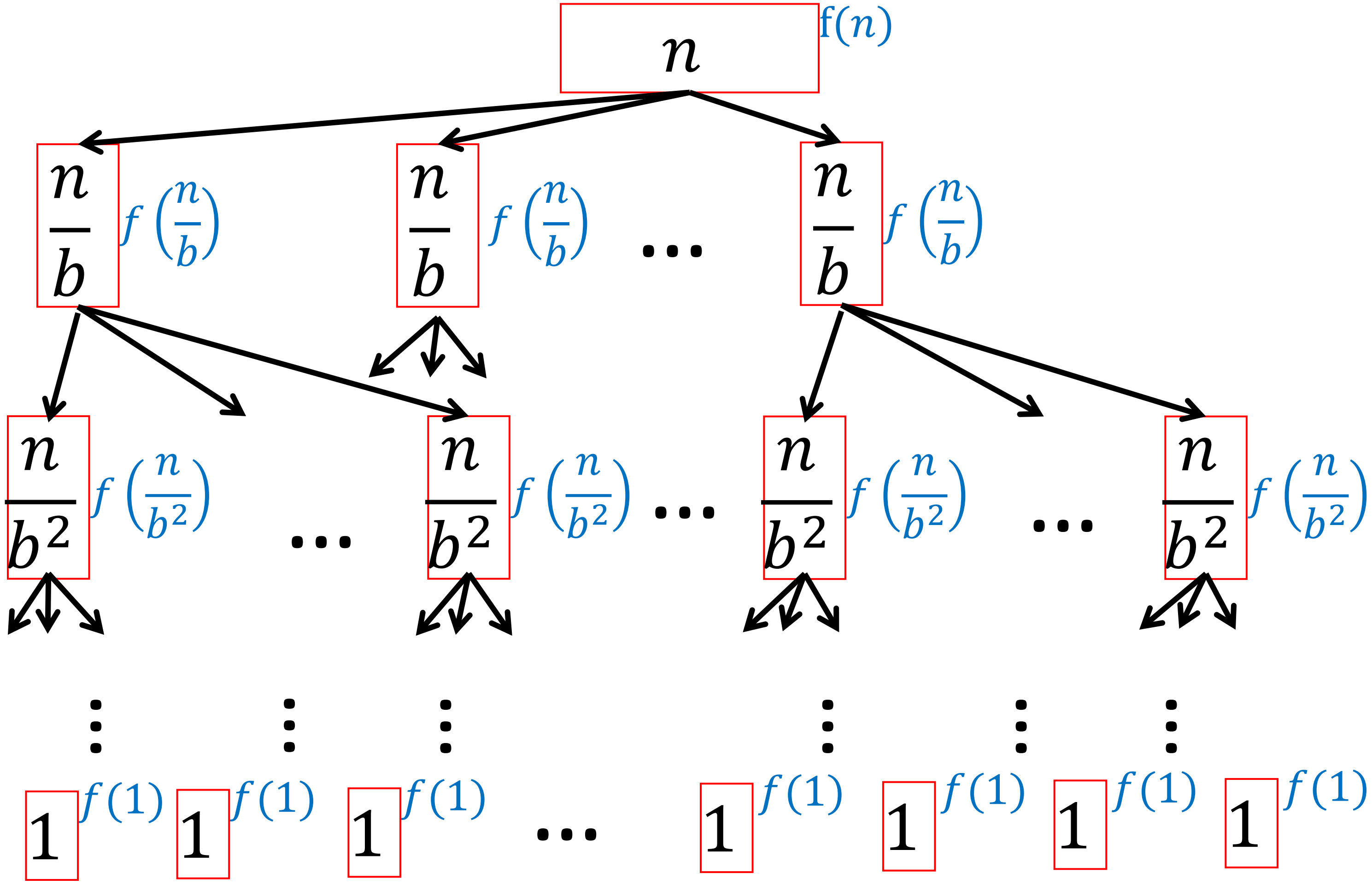
- **Divide**: $D(n)$ time,
- **Conquer**: recurse on small problems, size s
- **Combine**: $\mathrm{C}(n)$ time
- **Recurrence:**
  - $T(n) = D(n) + \sum T(s) + C(n)$

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

General

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

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



$f(n)$

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

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

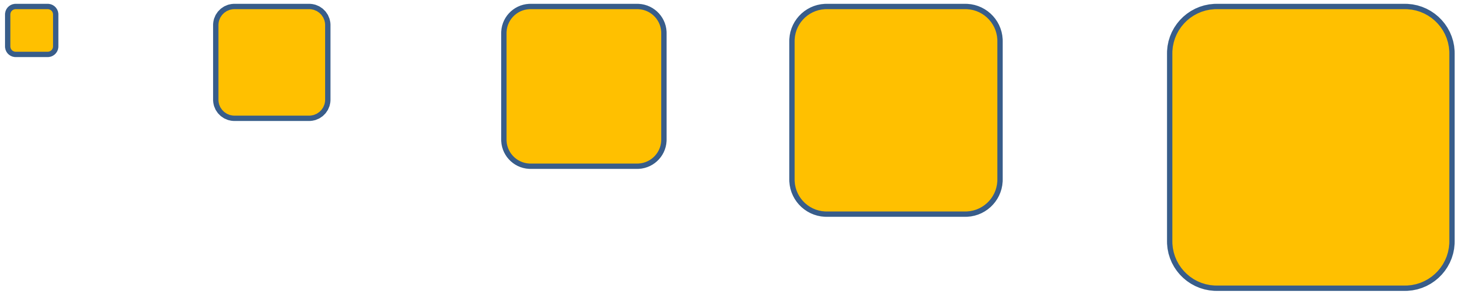
⋮

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

3 Cases

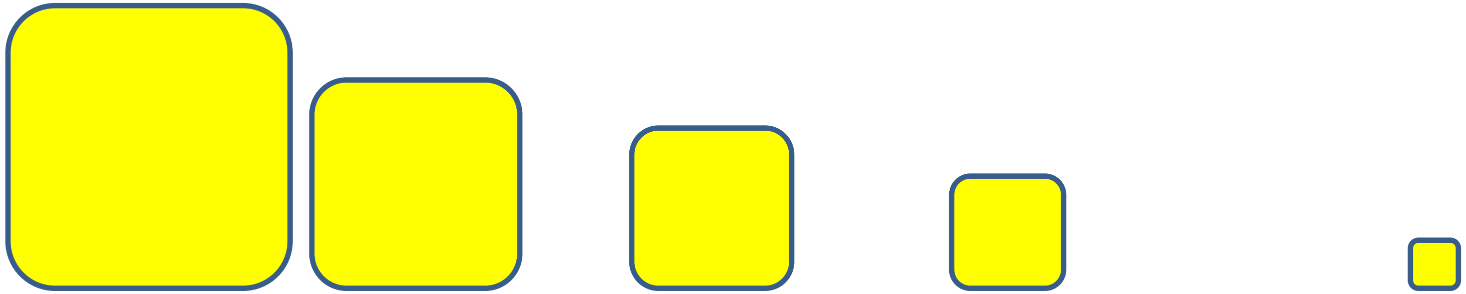
$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + a^3 f\left(\frac{n}{b^3}\right) + \cdots + a^L f\left(\frac{n}{b^L}\right)$$

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

Proof of Case 1

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

$$f(n) = O\left(n^{\log_b a - \varepsilon}\right) \Rightarrow f(n) \le c \cdot n^{\log_b n - \varepsilon}$$

Insert math here…

Conclusion: $T(n) = O(n^{\log_b a})$

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

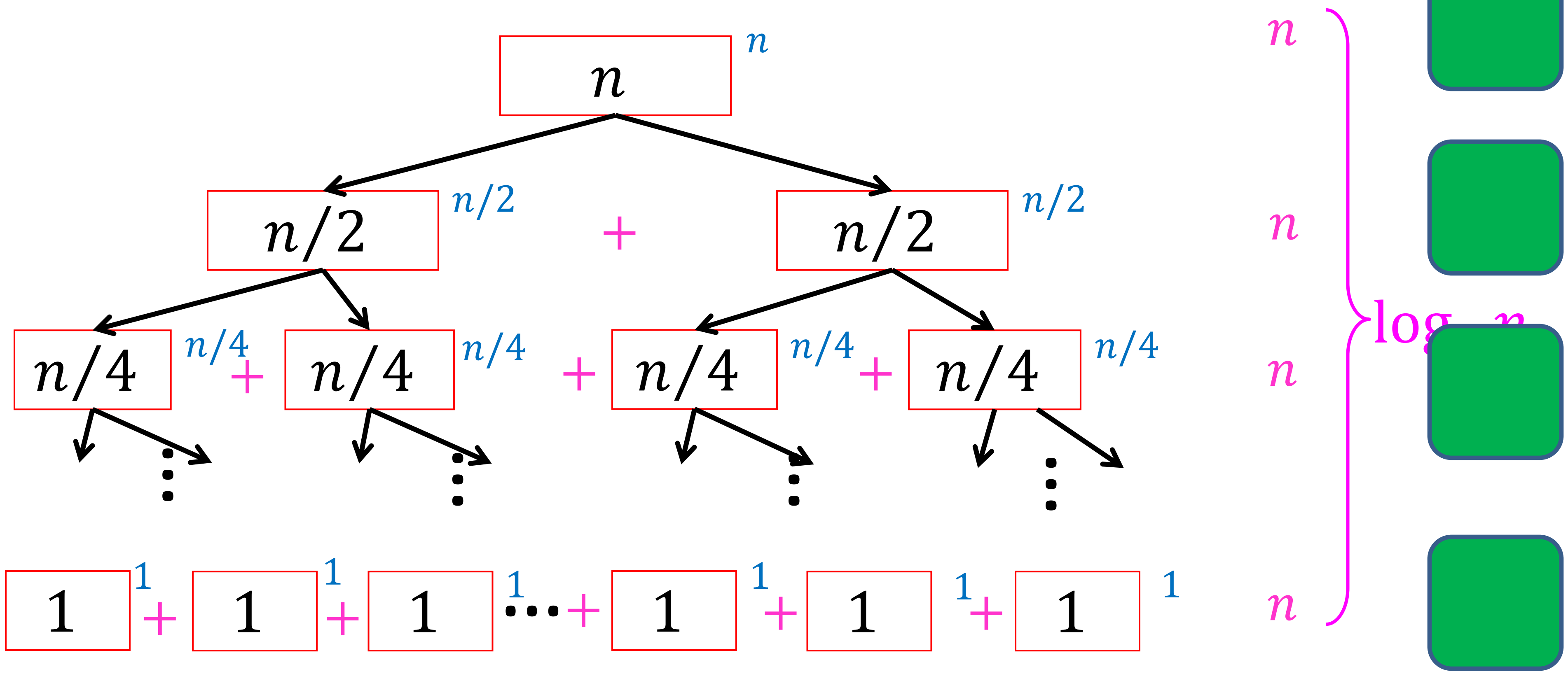
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

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

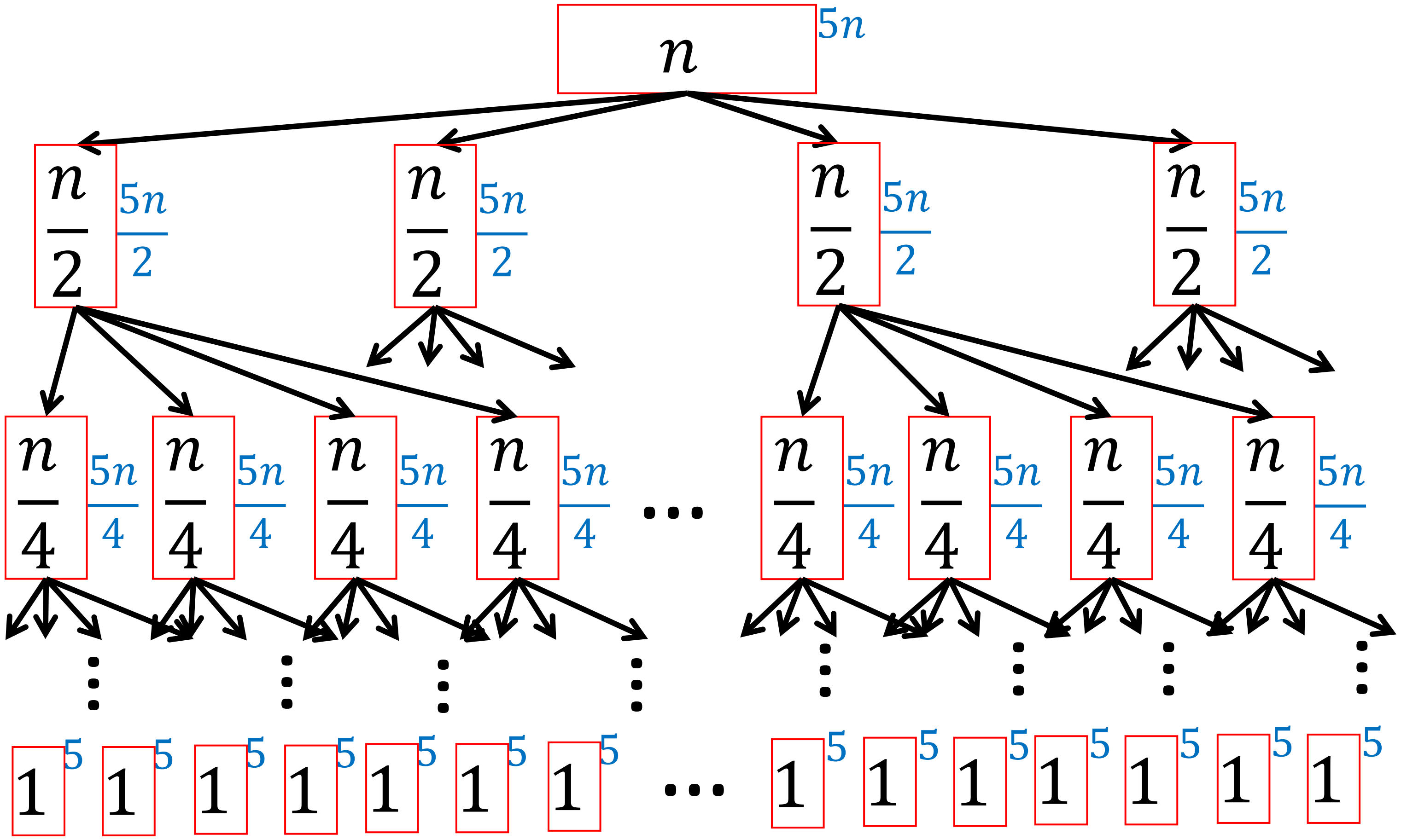
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Case 1

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

Tree method

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



$5n$

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

$\frac{1}{4}$

$\vdots$

$2^{\log_2}$

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

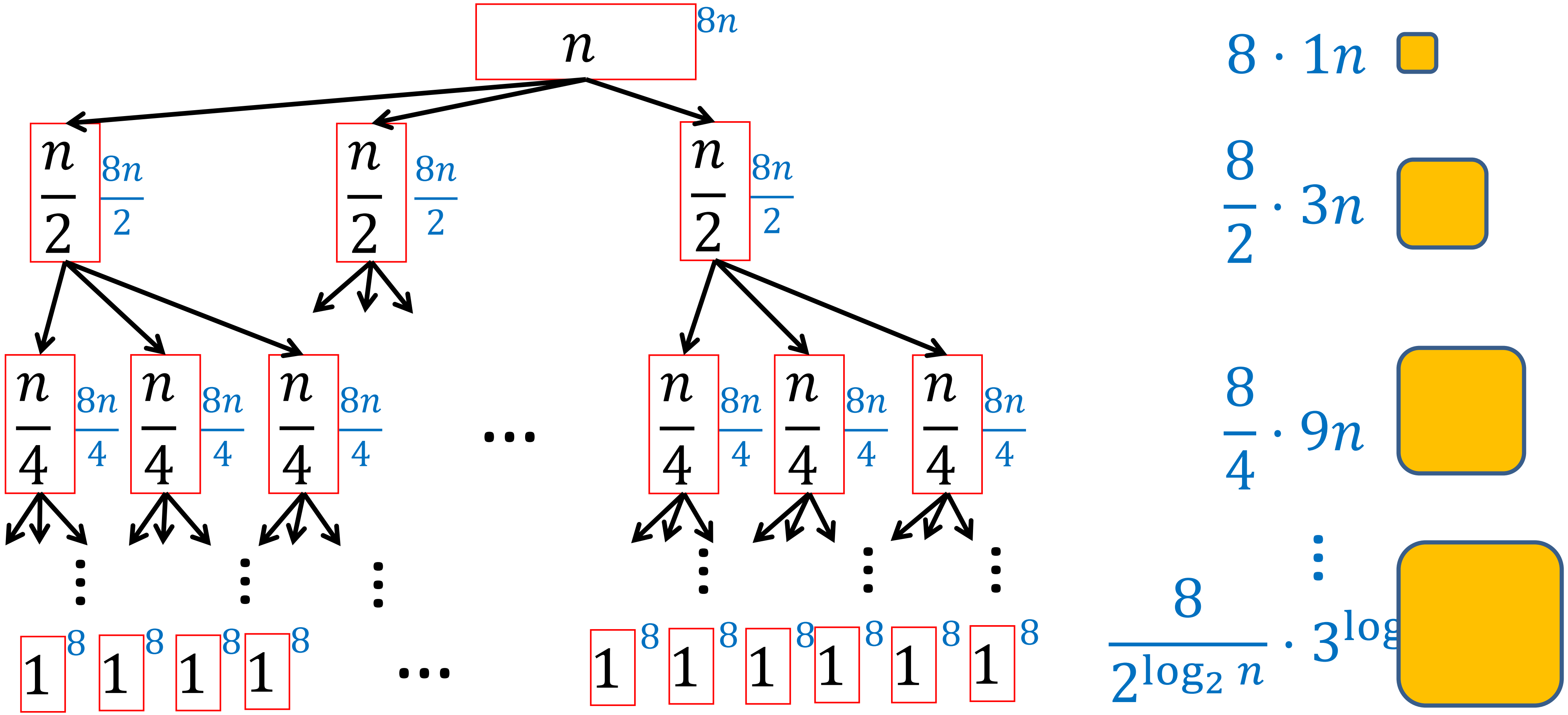
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

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

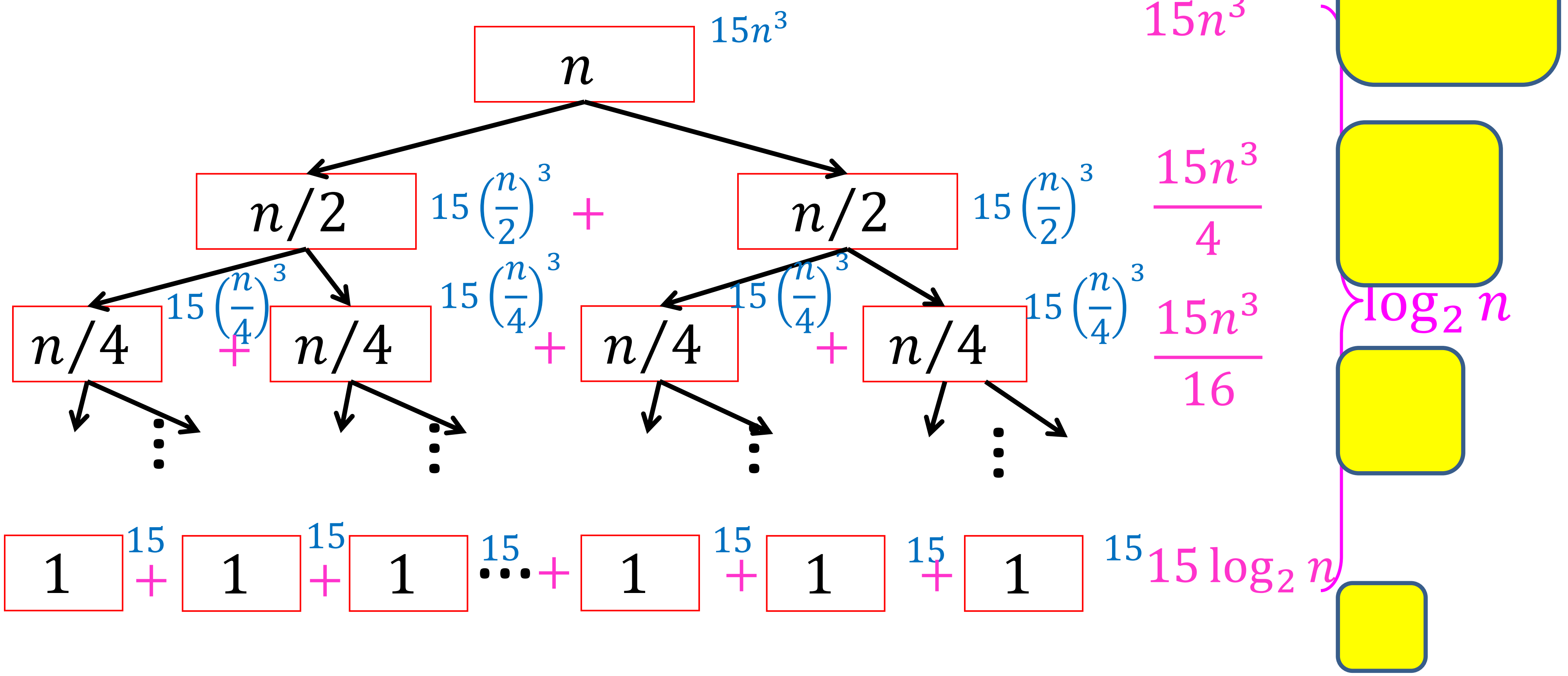
$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$

Case 3

$\Theta(n^3)$

Tree method

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



Homework Help Algorithm

- Algorithm: How to ask a question about homework (efficiently)
  1. Check to see if your question is already on piazza
  2. If it’s not on piazza, ask on piazza
  3. Look for other questions you know the answer to, and provide answers to any that you see
  4. TA office hours
  5. Instructor office hours
  6. Email, set up a meeting