CS 3100
Data Structures and Algorithms 2

Lecture 8: Divide and Conquer

Co-instructors: Robbie Hott and Ray Pettit

Spring 2024

Readings in CLRS 4th edition:

- Section 4.1-4.4

Announcements

- PS3 due tomorrow
- PA2 coming soon
- Office hours
  - Prof Hott Office Hours: Mondays 11a-12p, Fridays 10-11a and 2-3p
  - Prof Pettit Office Hours: Mondays and Wednesdays 2:30-4:00p
  - TA office hours posted on our website
- Quizzes 1-2 coming February 29, 2024
  - Both quizzes taken the same day
  - If you have SDAC, please schedule for 1 exam (*not a quiz*)

Divide and Conquer

[CLRS Chapter 4]

Divide:

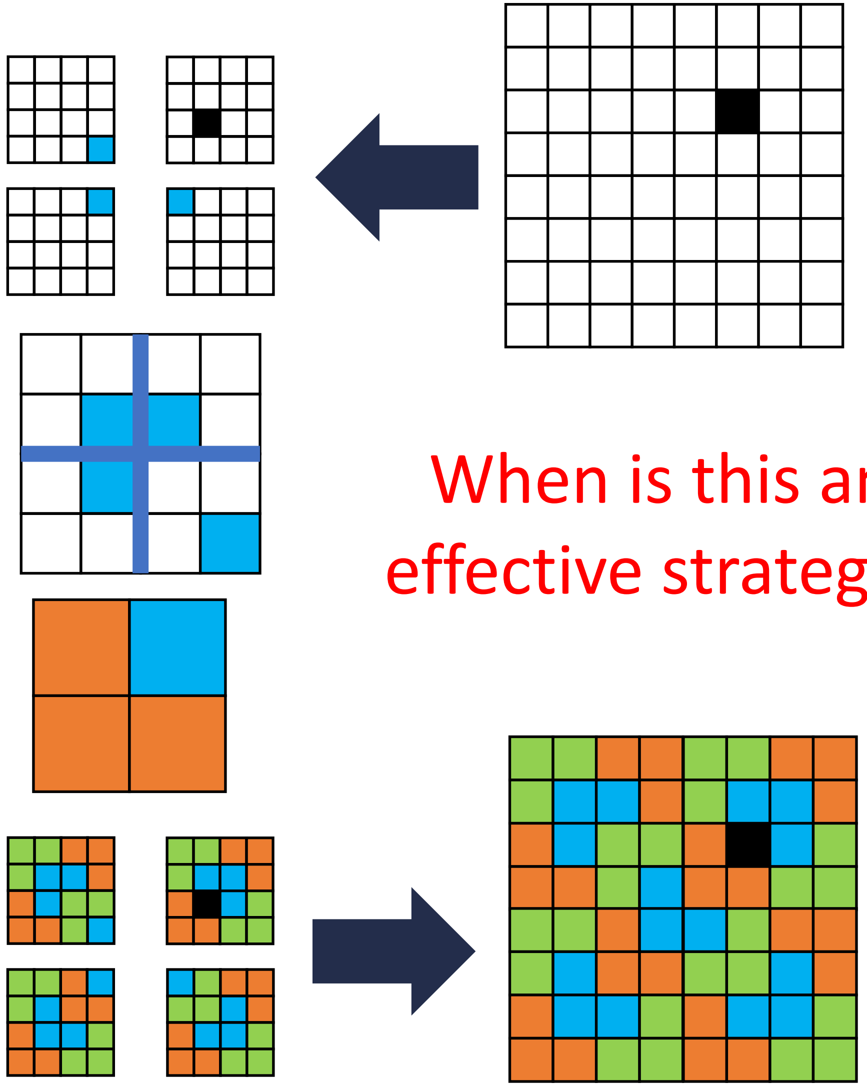
- Break the problem into multiple subproblems, each smaller instances of the original

Conquer:

- If the suproblems are "large":
  - Solve each subproblem recursively
- If the subproblems are "small":
  - Solve them directly (base case)

Combine:

- Merge solutions to subproblems to obtain solution for original problem



When is this an effective strategy?

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

How do we measure input size?

number of digits

What do we "count" for run time?

number of <u>elementary</u> operations
(single-digit multiplications)

"Schoolbook" Multiplication

How many multiplications?

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

n-digit numbers

n mults

n mults

n mults

n mults

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

What about cost of additions?

$\Theta(n^2)$

"Schoolbook" Multiplication

Can we do better?

How many 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 Multiplication

1. Break into smaller subproblems

$$\boxed{a}\,\boxed{b} = 10^{\frac{n}{2}}\boxed{a} + \boxed{b}$$

$$\times\,\boxed{c}\,\boxed{d} = 10^{\frac{n}{2}}\boxed{c} + \boxed{d}$$

$$= \quad 10^{n}\left(\boxed{a} \times \boxed{c}\right) +$$

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

$$\left(\boxed{b} \times \boxed{d}\right)$$

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:

- $10^n(ac) + 10^{n/2}(ad + bc) + bd$

For simplicity, assume that $n = 2^k$ is a power of 2

Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

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

Recursively solve

$$T(n)$$

Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

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

Recursively solve

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

Need to compute 4 multiplications, each of size $n/2$

Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

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

Recursively solve

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

Need to compute 4 multiplications, each of size $n/2$

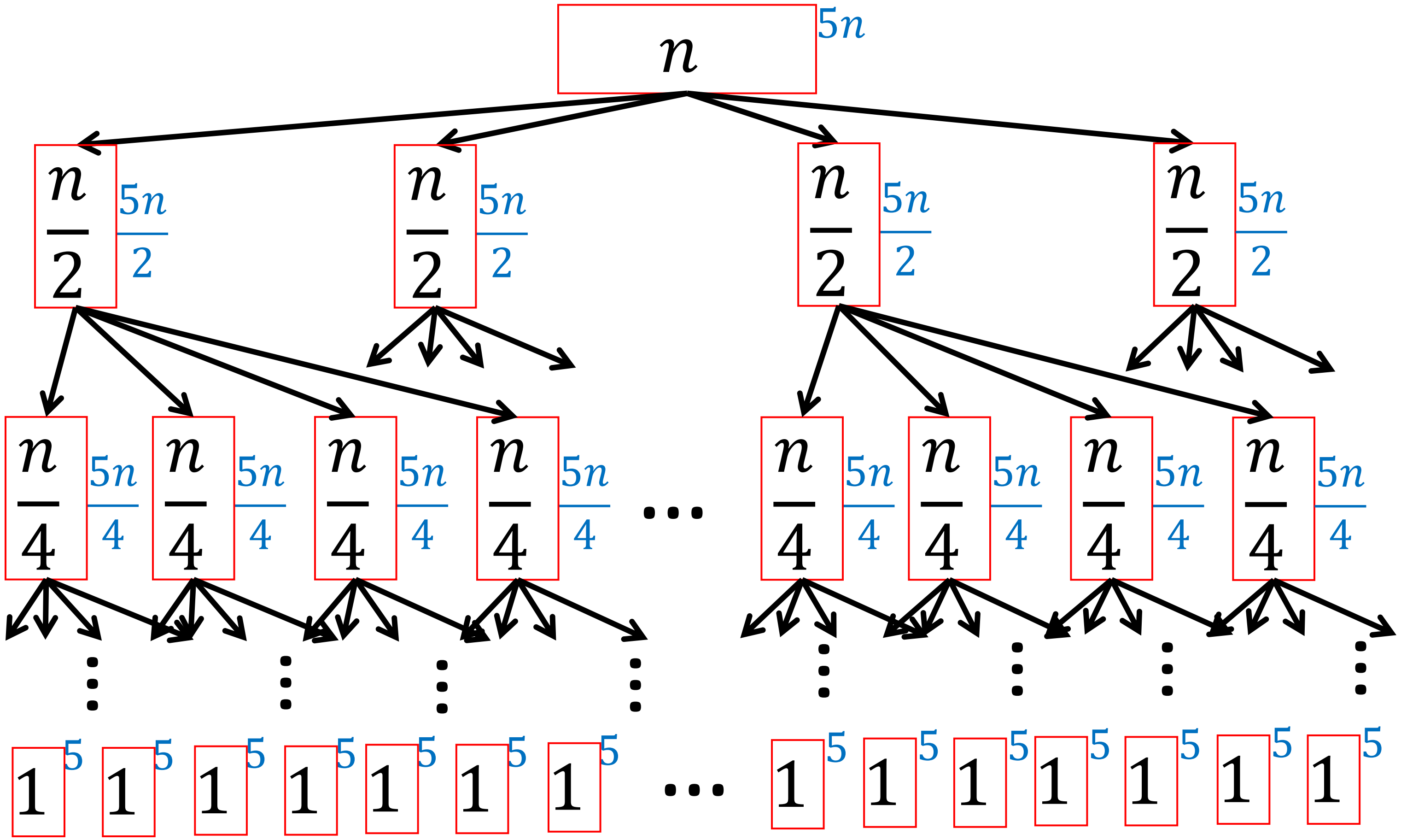
2 shifts and 3 additions on n-bit values

Divide and Conquer Multiplication

3. Use asymptotic notation to simplify

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

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



$$5n$$

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

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

$$\vdots$$

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

Divide and Conquer Multiplication

3. Use asymptotic notation to simplify

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

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

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

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

No better than the schoolbook method!

Karatsuba Multiplication

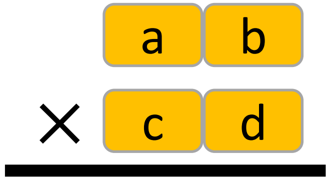
1. Break into smaller subproblems

$$
\begin{array}{rcl}
 & \boxed{a}\,\boxed{b} & = 10^{\frac{n}{2}}\,\boxed{a} + \boxed{b} \\
\times & \boxed{c}\,\boxed{d} & = 10^{\frac{n}{2}}\,\boxed{c} + \boxed{d}
\end{array}
$$

$$
\begin{aligned}
= \quad & 10^{n}\left(\boxed{a} \times \boxed{c}\right) + \\
& 10^{\frac{n}{2}}\left(\boxed{a} \times \boxed{d} + \boxed{b} \times \boxed{c}\right) + \\
& \left(\boxed{b} \times \boxed{d}\right)
\end{aligned}
$$

Recall: previous divide-and-conquer recursively computed ac, ad, bc, bd

Karatsuba Multiplication



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

Can’t avoid these

This can be simplified!

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

$$\boxed{ac}+\boxed{ad+bc}+\boxed{bd}$$

$$\boxed{ad+bc}=\boxed{(a+b)(c+d)-\boxed{ac}-\boxed{bd}}$$

Two multiplications

One multiplication

Karatsuba Multiplication

2. Use recurrence relation to express recursive running time

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

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

Recursively solve

$$T(n) =$$

Karatsuba Multiplication

	a	b
×	c	d

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

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

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

Karatsuba Multiplication

2. Use recurrence relation to express recursive running time

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

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

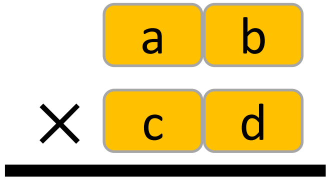
Recursively solve

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

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

2 shifts and 6 additions on n-bit values

Karatsuba 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, bd, $(a+b)(c+d)$
- If $n = 1$:
  - Compute ac, bd, $(a+b)(c+d)$ directly (base case)

Combine:

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

Karatsuba Multiplication

$$\begin{array}{rcc} & a & b \\ \times & c & d \\ \hline \end{array}$$

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

<u>Pseudocode:</u>

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

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

Karatsuba Multiplication

$$\begin{array}{rcc} & a & b \\ \times & c & d \\ \hline \end{array}$$

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

<u>Pseudocode:</u>

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

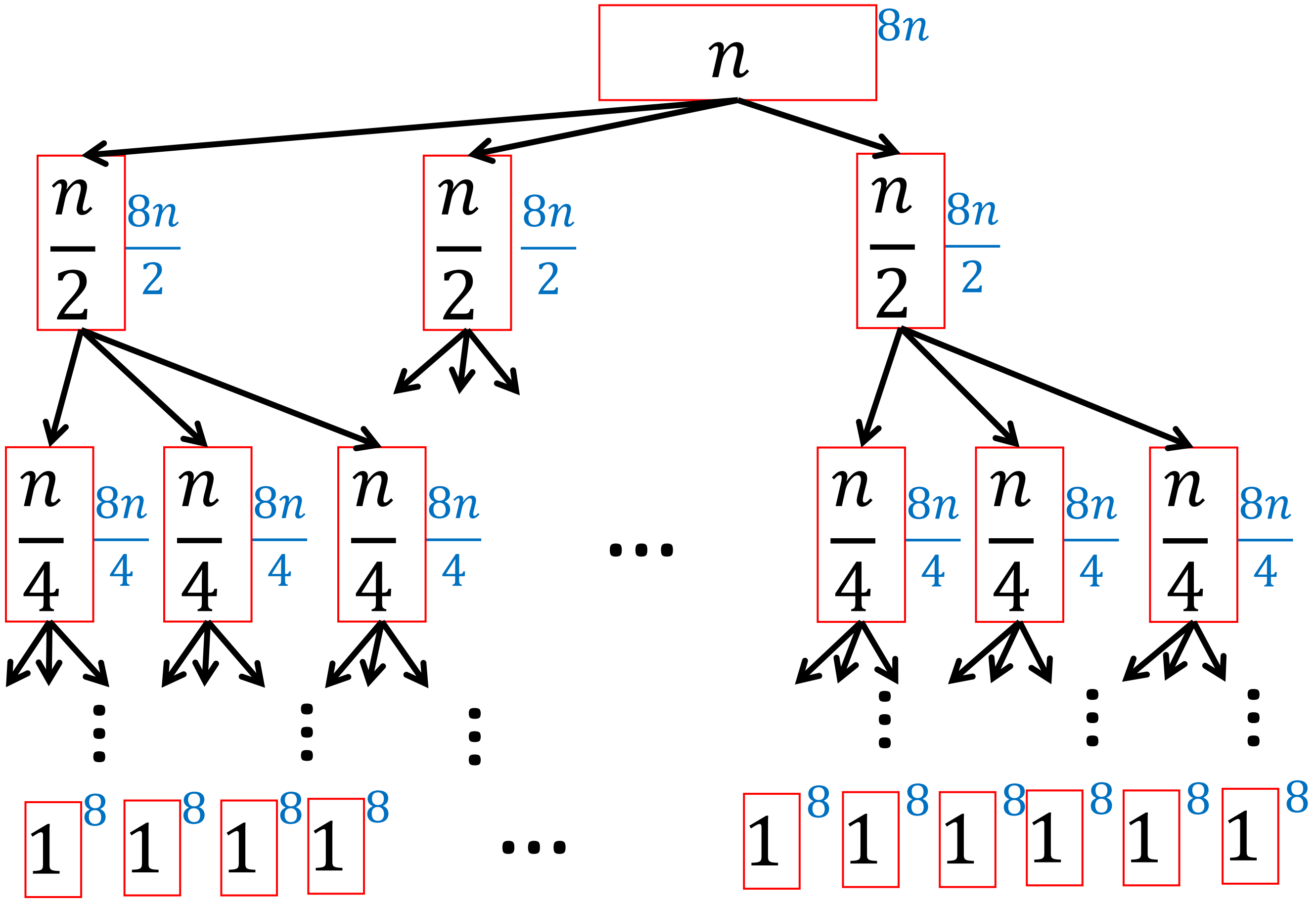
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Karatsuba

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…(on board, see lecture supplement)

Karatsuba

Karatsuba

Karatsuba Multiplication

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

How to simplify this
(using asymptotic notation)?

Drop constant multiples

Karatsuba Multiplication

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

How to simplify this (using asymptotic notation)?

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

Drop constant multiples

$$= \Theta\left(\frac{3}{2}n\cdot\left({}^{3}/_{2}\right)^{\log_2 n}-n\right)$$

Distribute terms

Karatsuba Multiplication

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

How to simplify this (using asymptotic notation)?

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

Drop constant multiples

$$= \Theta\left(\frac{3}{2}n \cdot \left({}^{3}/_{2}\right)^{\log_2 n} - n\right)$$

Distribute terms

$$= \Theta\left(n \cdot \left({}^{3}/_{2}\right)^{\log_2 n}\right)$$

Drop constants and low-order terms

Karatsuba Multiplication

$$T(n) = \Theta\left(n \cdot \left({}^{3}/_{2}\right)^{\log_2 n}\right)$$

How to simplify this
(using asymptotic notation)?

Properties of logarithms:

$$2^{\log_2 n} = n$$

$$a^{bc} = \left(a^b\right)^c$$

$$3^{\log_2 n} = 2^{\log_2\left(3^{\log_2 n}\right)} = 2^{(\log_2 n)(\log_2 3)} = \left(2^{\log_2 n}\right)^{\log_2 3} = n^{\log_2 3}$$

$$2^{\log_2 n} = n$$

$$\log a^b = b \log a$$

$$2^{\log_2 n} = n$$

Karatsuba Multiplication

$$T(n) = \Theta\left(n \cdot \left(3/_2\right)^{\log_2 n}\right)$$

$$= \Theta\left(n \cdot \left(\frac{3^{\log_2 n}}{2^{\log_2 n}}\right)\right)$$

$$= \Theta\left(n \cdot \left(\frac{n^{\log_2 3}}{n}\right)\right)$$

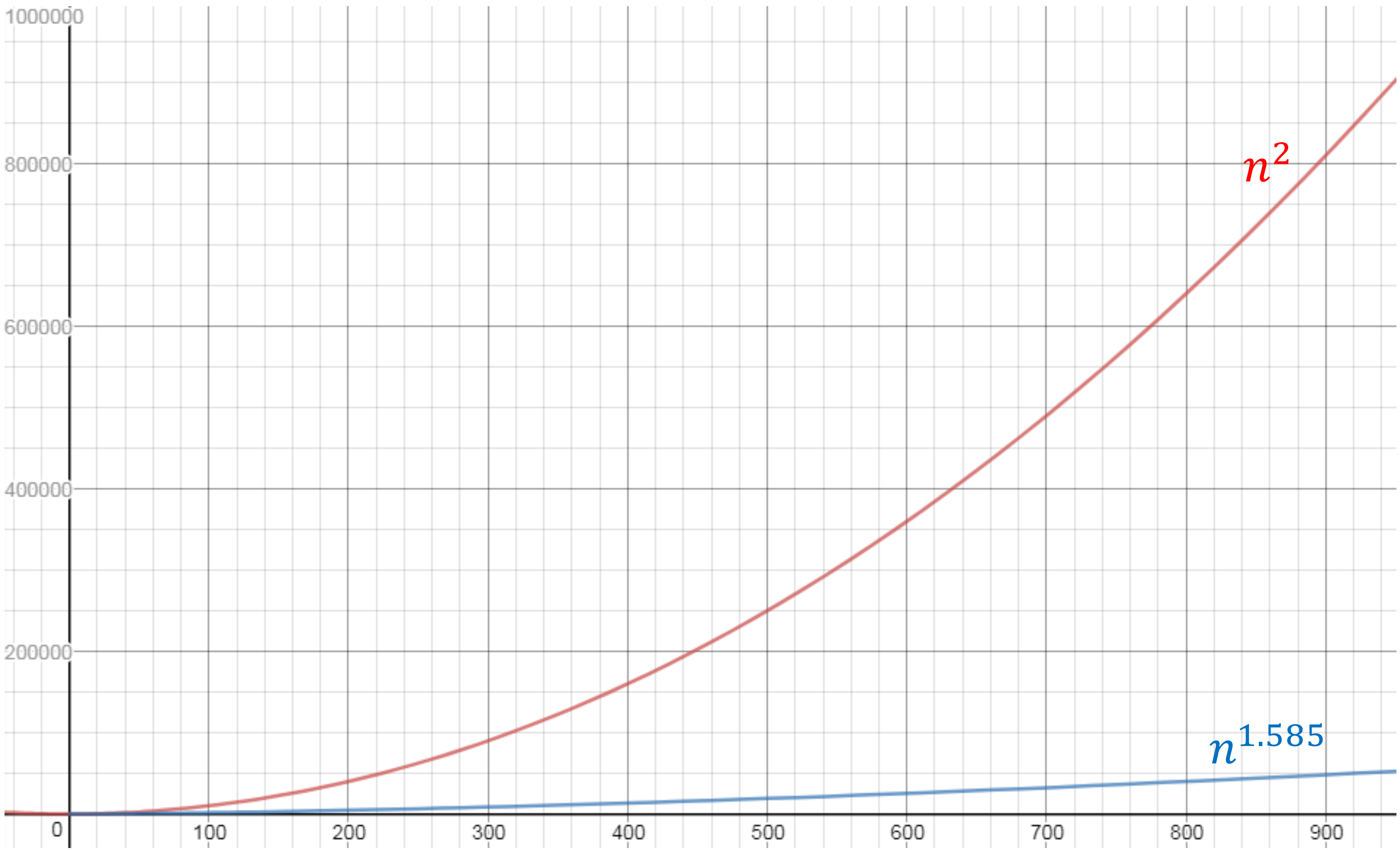
$$= \Theta\left(n^{\log_2 3}\right) \approx \Theta\left(n^{1.585}\right)$$

How to simplify this (using asymptotic notation)?

$$2^{\log_2 n} = n$$

$$3^{\log_2 n} = n^{\log_2 3}$$

Strictly better than schoolbook method!


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

Analyzing Divide and Conquer

1. Break into smaller subproblems
2. Use recurrence relation to express recursive running time
3. Use asymptotic notation to simplify

Divide: $D(n)$ time

Conquer: Recurse on smaller problems of size $s_1, \dots, s_k$

Combine: $C(n)$ time

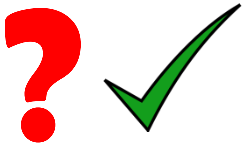
Recurrence:

- $T(n) = D(n) + \sum_{i \in [k]} T(s_i) + C(n)$

Recurrence Solving Techniques

Tree

get a picture of recursion



Guess/Check

guess and use induction to prove

"Cookbook"

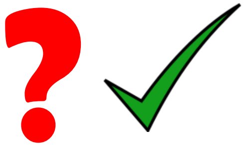
MAGIC!

Substitution

substitute in to simplify

Recurrence Solving Techniques

Tree



Guess/Check (induction)

"Cookbook"

Substitution

Guess and Check Blueprint

Show: $T(n) = O(g(n))$

Consider: $g_*(n) = c \cdot g(n)$ for some constant c

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 $T(1) \leq g_*(1)$ (sometimes, may need to consider additional base cases)
- Hypothesis:
  - $\forall n \leq x_0, T(n) \leq g_*(n)$
- Inductive step:
  - Show that $T(x_0 + 1) \leq g_*(x_0 + 1)$

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

Mergesort Guess and Check

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

Karatsuba Analysis using Guess and Check

$$T(n) = 3T(n/2) + 8n$$

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

Base case: $T(1) = 8 \leq 3000$

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

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

Karatsuba Guess and Check (Loose)

$$T(n) = 3T(n/2) + 8n$$

Hypothesis: $\forall n \le x_0\colon\ T(n) \le 3000n^{1.6}$

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

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

Recurrence definition

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

Inductive hypothesis

Karatsuba Guess and Check (Loose)

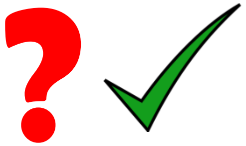
Karatsuba Guess and Check (Loose)

$T(x_0+1)$	$= 3T\left(\frac{x_0+1}{2}\right) + 8(x_0+1)$	Recurrence definition
	$\leq 3\left(3000\left(\frac{x_0+1}{2}\right)^{1.6}\right) + 8(x_0+1)$	Inductive hypothesis
	$\leq 3\left(3000\left(\frac{x_0+1}{2}\right)^{1.6}\right) + 8(x_0+1)^{1.6}$	$\forall x \geq 0: x^{1.6} \geq x$
	$= \left(\frac{9000}{2^{1.6}} + 8\right)(x_0+1)^{1.6}$	Distributive property
	$\leq 3000(x_0+1)^{1.6}$	$\frac{9000}{2^{1.6}} + 8 \leq 3000$

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

Recurrence Solving Techniques

Tree



Guess/Check

“Cookbook”

Substitution

Observation

Divide: $D(n)$ time

Conquer: Recurse on smaller problems of size $s_1, \ldots, s_k$

Combine: $C(n)$ time

Recurrence:

- $T(n) = D(n) + \sum_{i \in [k]} T(s_i) + C(n)$

Many divide and conquer algorithms have recurrences are of form:

- $T(n) = a \cdot T(n/b) + f(n)$

a and b are constants

Mergesort: $T(n) = 2T(n/2) + n$

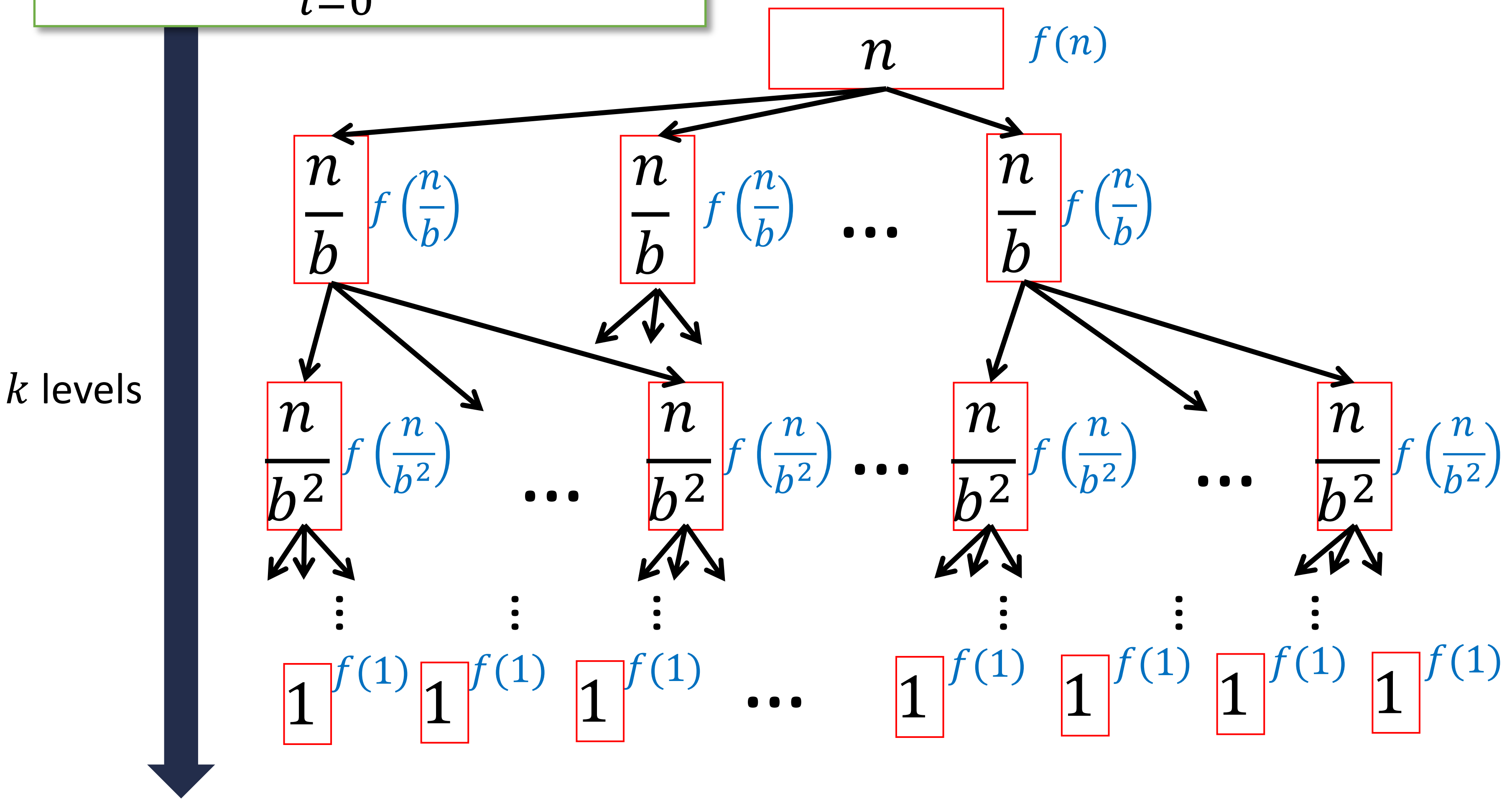
Divide and Conquer Multiplication: $T(n) = 4T(n/2) + 5n$

Karatsuba Multiplication: $T(n) = 3T(n/2) + 8n$

General Recurrence

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

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



Number of subproblems	Cost of subproblem
1	$f(n)$
a	$f(n/b)$
a^2	$f(n/b^2)$
a^k	$f(n/b^k)$

General Recurrence

$k = \log_b n$

An aside:

$a^{\log_b n} =$

Three Cases

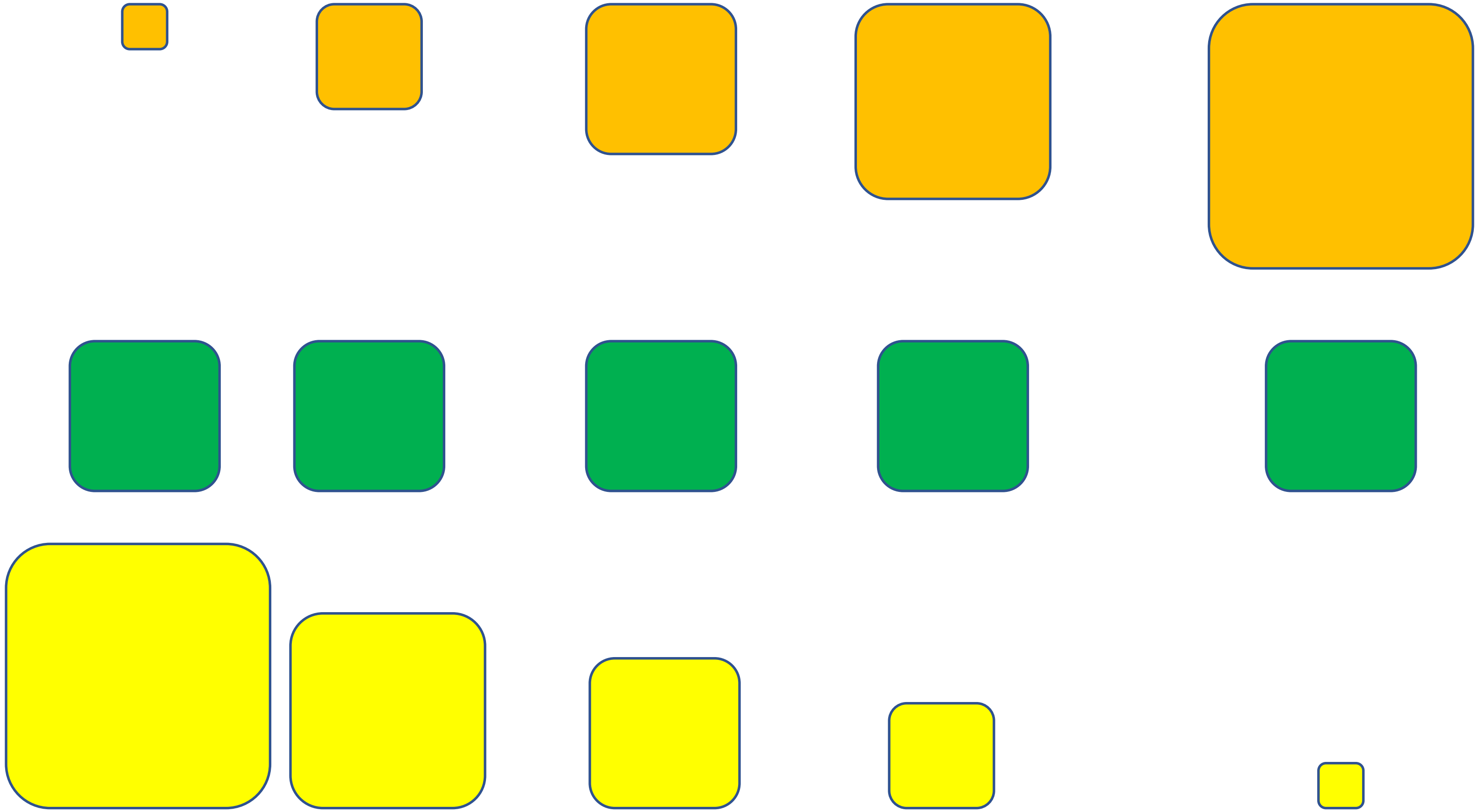
$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + a^3 f\left(\frac{n}{b^3}\right) + \cdots + a^k f\left(\frac{n}{b^k}\right)$$

$$k = \log_b n$$

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

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

$$\delta = \log_b a$$

	Requirement on f	Implication
Case 1	$f(n) \in O(n^{\delta-\varepsilon})$ for some constant $\varepsilon > 0$	$T(n) \in \Theta(n^{\delta})$

Master Theorem

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

$$\delta = \log_b a$$

	Requirement on f	Implication
Case 1	$f(n) \in O\left(n^{\delta-\varepsilon}\right)$ for some constant $\varepsilon > 0$	$T(n) \in \Theta\left(n^{\delta}\right)$
Case 2	$f(n) \in \Theta\left(n^{\delta}\right)$	$T(n) \in \Theta\left(n^{\delta}\log n\right)$

Master Theorem

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

$$\delta = \log_b a$$

	Requirement on f	Implication
Case 1	$f(n) \in O\left(n^{\delta-\varepsilon}\right)$ for some constant $\varepsilon > 0$	$T(n) \in \Theta\left(n^{\delta}\right)$
Case 2	$f(n) \in \Theta\left(n^{\delta}\right)$	$T(n) \in \Theta\left(n^{\delta} \log n\right)$
Case 3	$f(n) \in \Omega\left(n^{\delta+\varepsilon}\right)$ for some constant $\varepsilon > 0$ **AND** $af\left(\frac{n}{b}\right) \leq cf(n)$ for constant $c < 1$ and sufficiently large n	$T(n) \in \Theta\left(f(n)\right)$

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

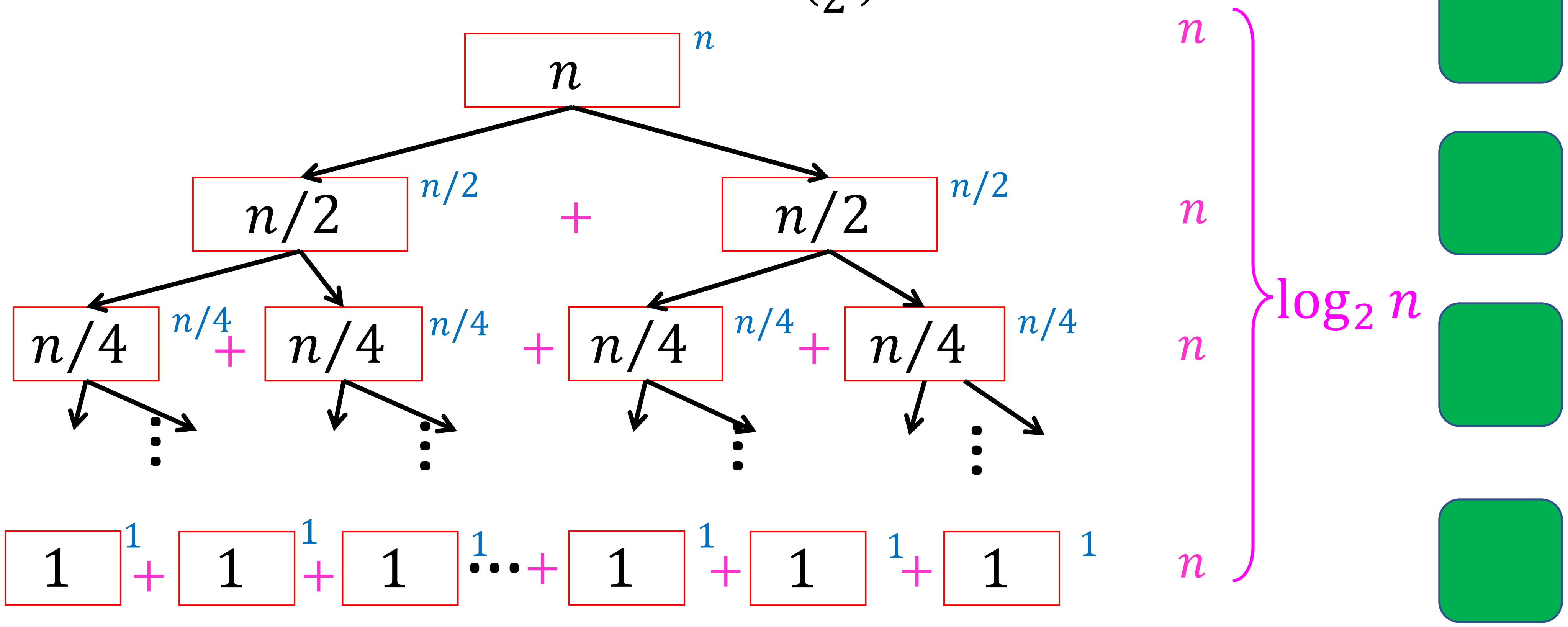
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

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

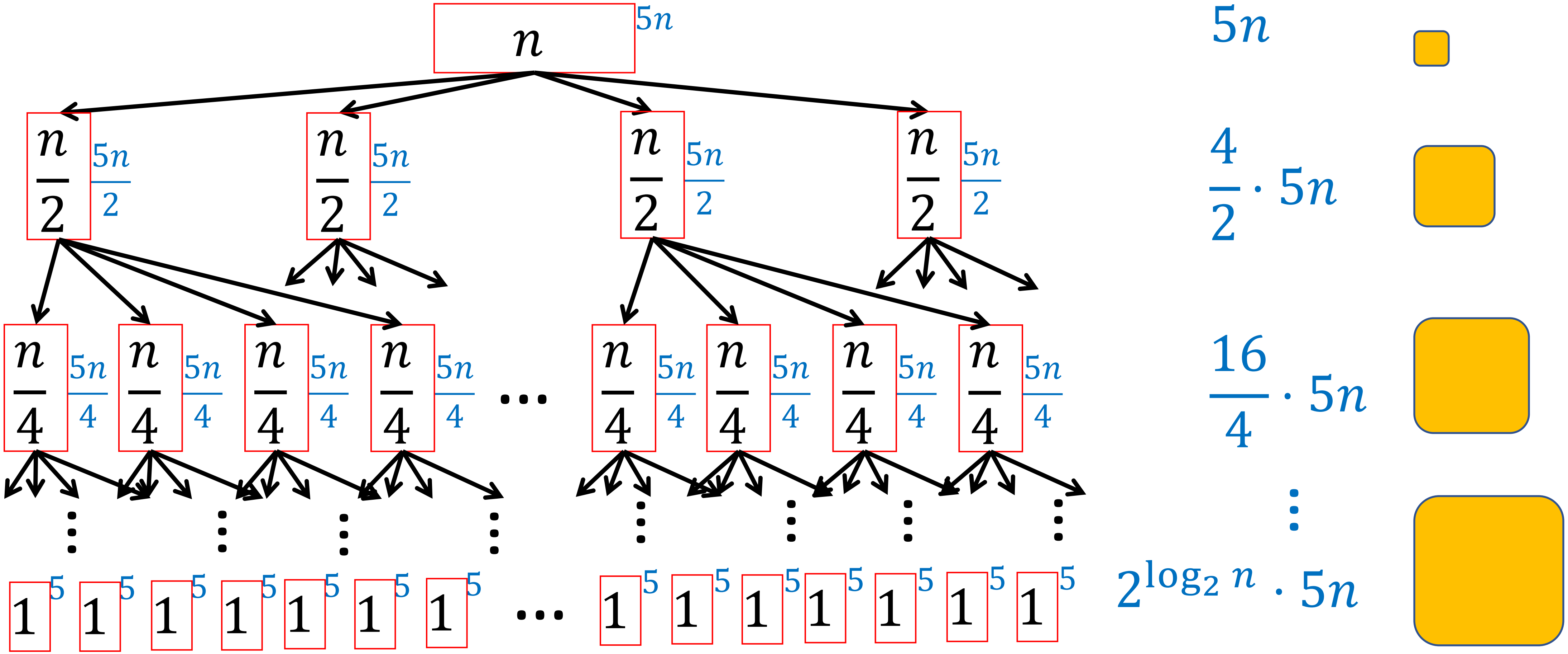
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Case 1

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

Tree method

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



Tree method

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

Cost is <u>increasing</u> with the recursion depth (due to large number of subproblems)

Most of the work happening in the leaves

$$5n$$

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

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

$$\vdots$$

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

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

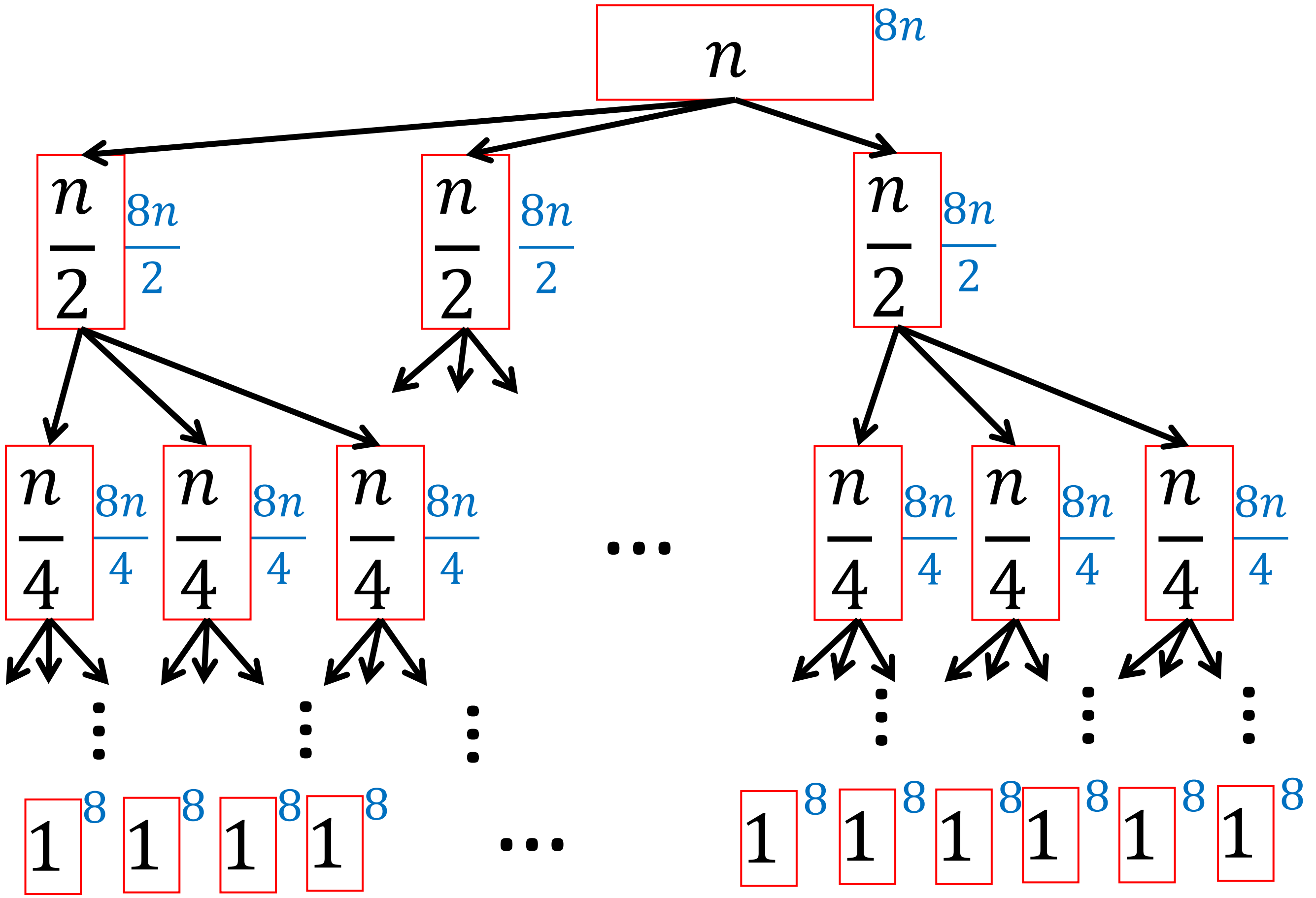
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Case 1

$$\Theta\left(n^{\log_2 3}\right) \approx \Theta(n^{1.585})$$

Karatsuba

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



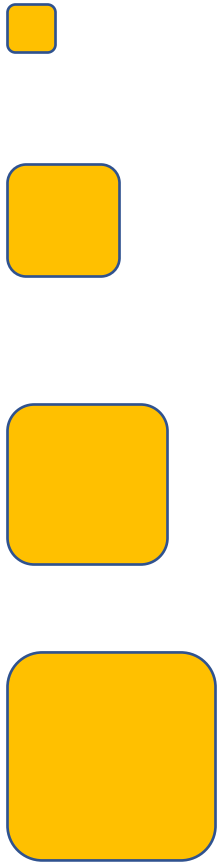
$8 \cdot 1n$

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

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

⋮

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



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

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

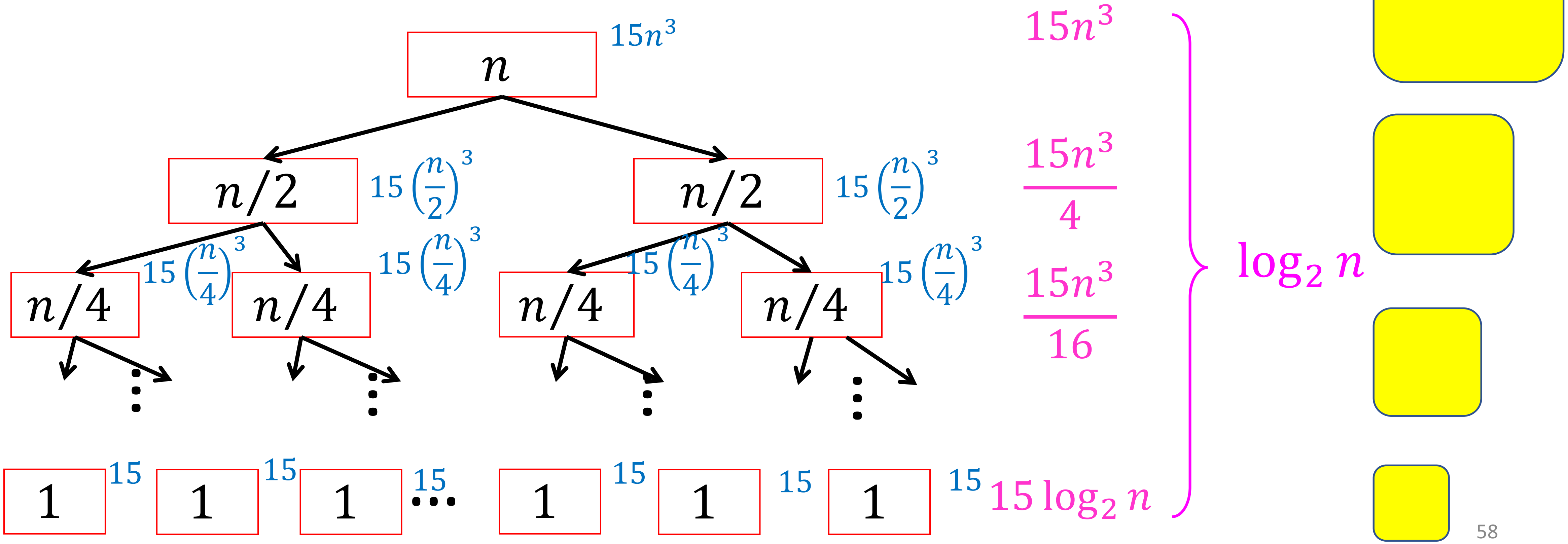
$\Theta(n^3)$

Important: For Case 3, need to additionally check that $2f(n/2) \leq cf(n)$ for constant $c < 1$ and sufficiently large n

$$2f(n/2) = 30(n/2)^3 = \frac{30}{8}n^3 \leq \frac{1}{4}(15n^3)$$

Master Theorem Example 4 (Visually)

$$T(n) = 2T(n/2) + 15n^3$$



Master Theorem Example 4 (Visually)

$$T(n) = 2T(n/2) + 15n^3$$

Cost is <u>decreasing</u> with the recursion depth (due to high *non-recursive* cost)

Most of the work happening at the top

$$\left.\begin{array}{c} 15n^3 \\[2em] \dfrac{15n^3}{4} \\[1em] \dfrac{15n^3}{16} \\[3em] 15\log_2 n \end{array}\right\} \log_2 n$$