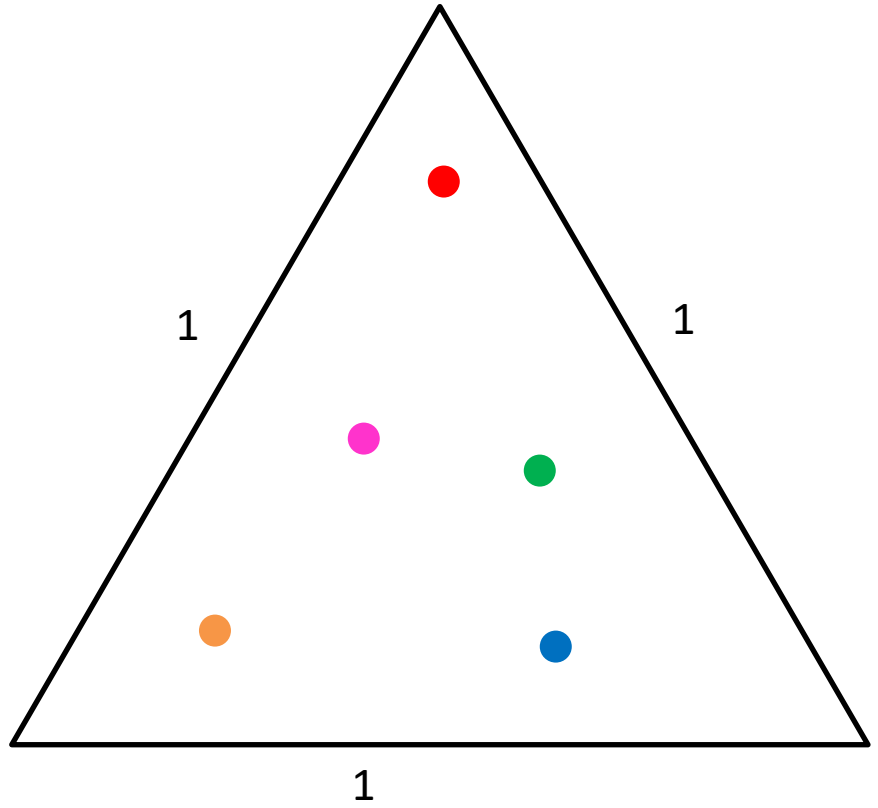
CS4102 Algorithms

Spring 2020

Warm up

Given 5 points on the unit equilateral triangle, show there's always a pair of distance $\leq \frac{1}{2}$ apart



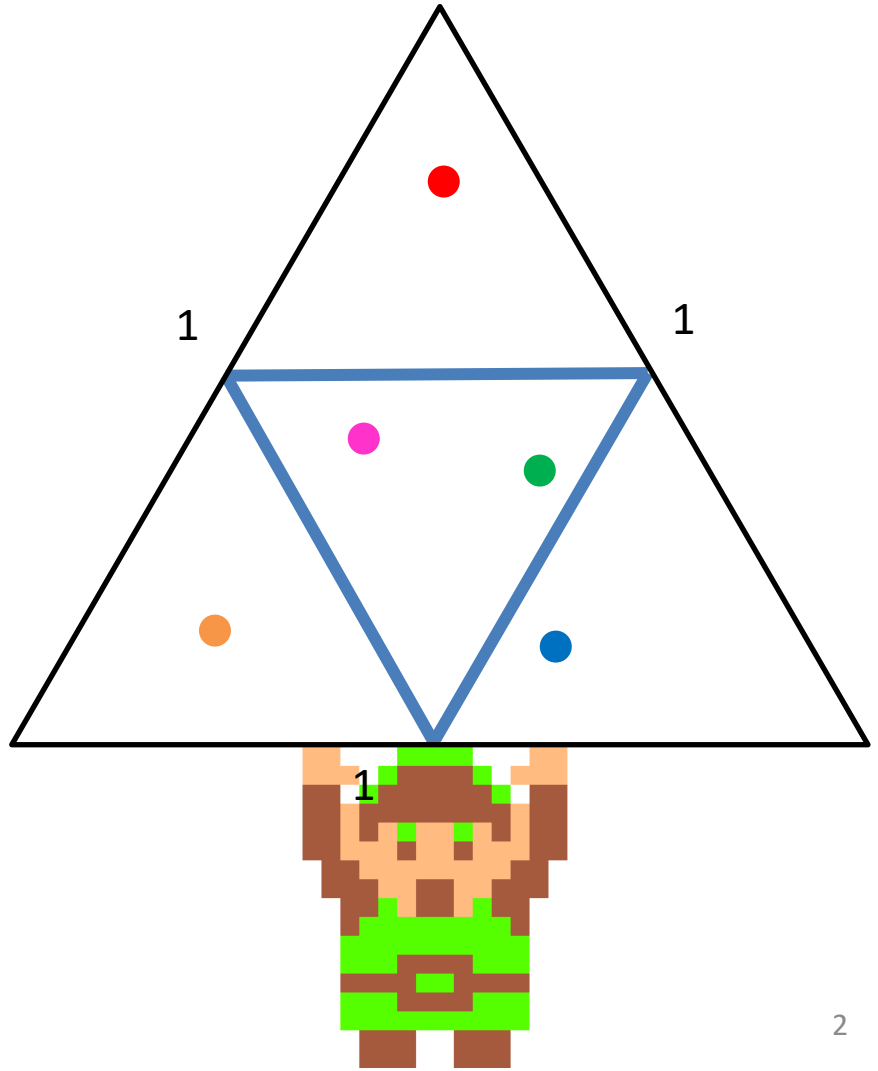
CS4102 Algorithms

Spring 2020

If points p_1, p_2 in same quadrant, then $\delta(p_1, p_2) \leq \frac{1}{2}$

Given 5 points, two must share the same quadrant

Pigeonhole Principle!



Today's Keywords

- Divide and Conquer
- Closest Pair of Points

CLRS Readings

- Chapter 4

Homeworks

- HW2 due Thursday 2/6 at 11pm
  - Written (use Latex!) – Submit BOTH pdf and zip!
  - Asymptotic notation
  - Recurrences
  - Master Theorem
  - Divide and Conquer

Recurrence Solving Techniques

Tree

Guess/Check

"Cookbook"

Substitution

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

3 Cases

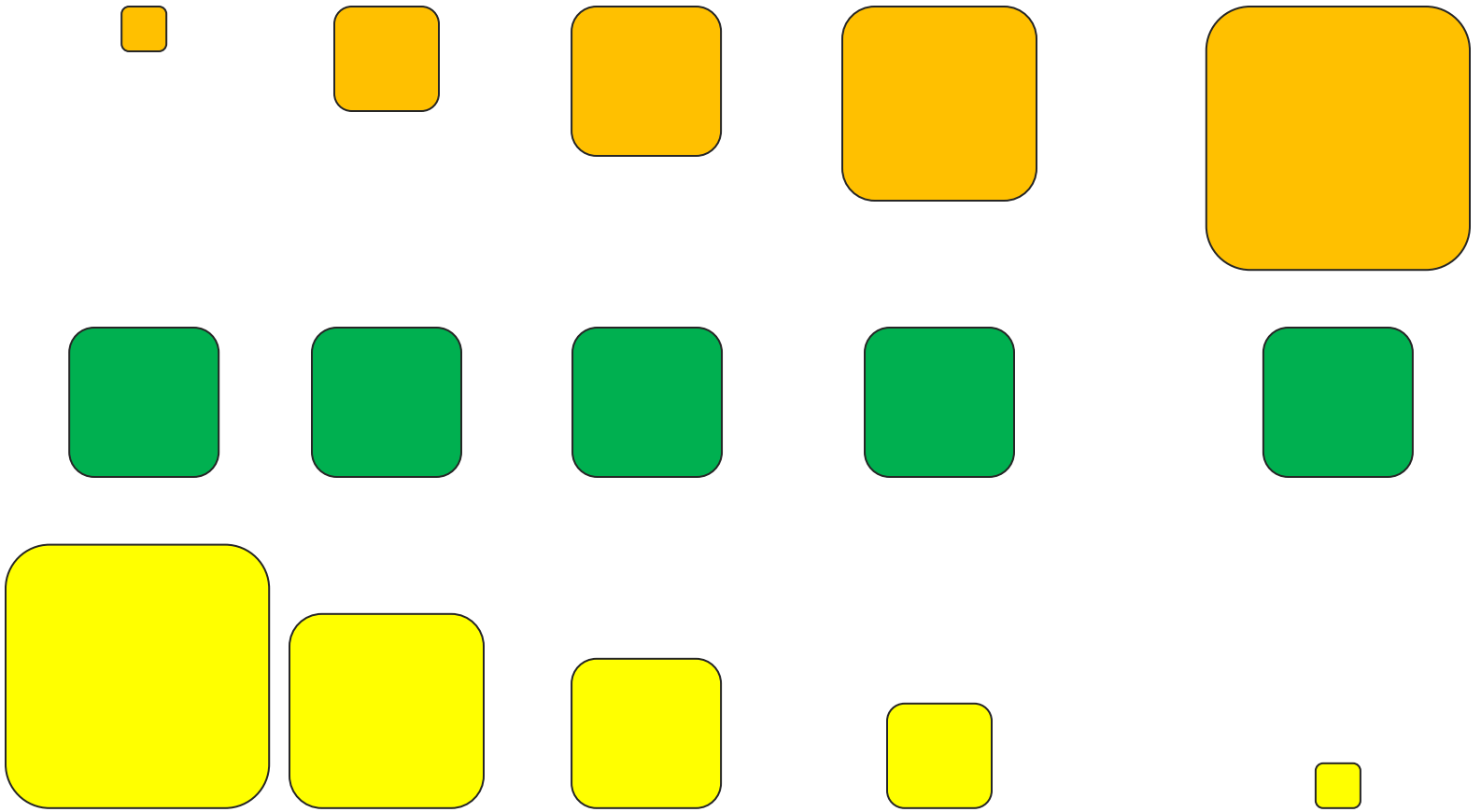
$L = \log_b n$

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

Case 1:
Most work happens at the leaves

Case 2:
Work happens consistently throughout

Case 3:
Most work happens at top of tree



Historical Aside: Master Theorem

Jon Bentley

Dorothea Haken

No Picture Found

James Saxe

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

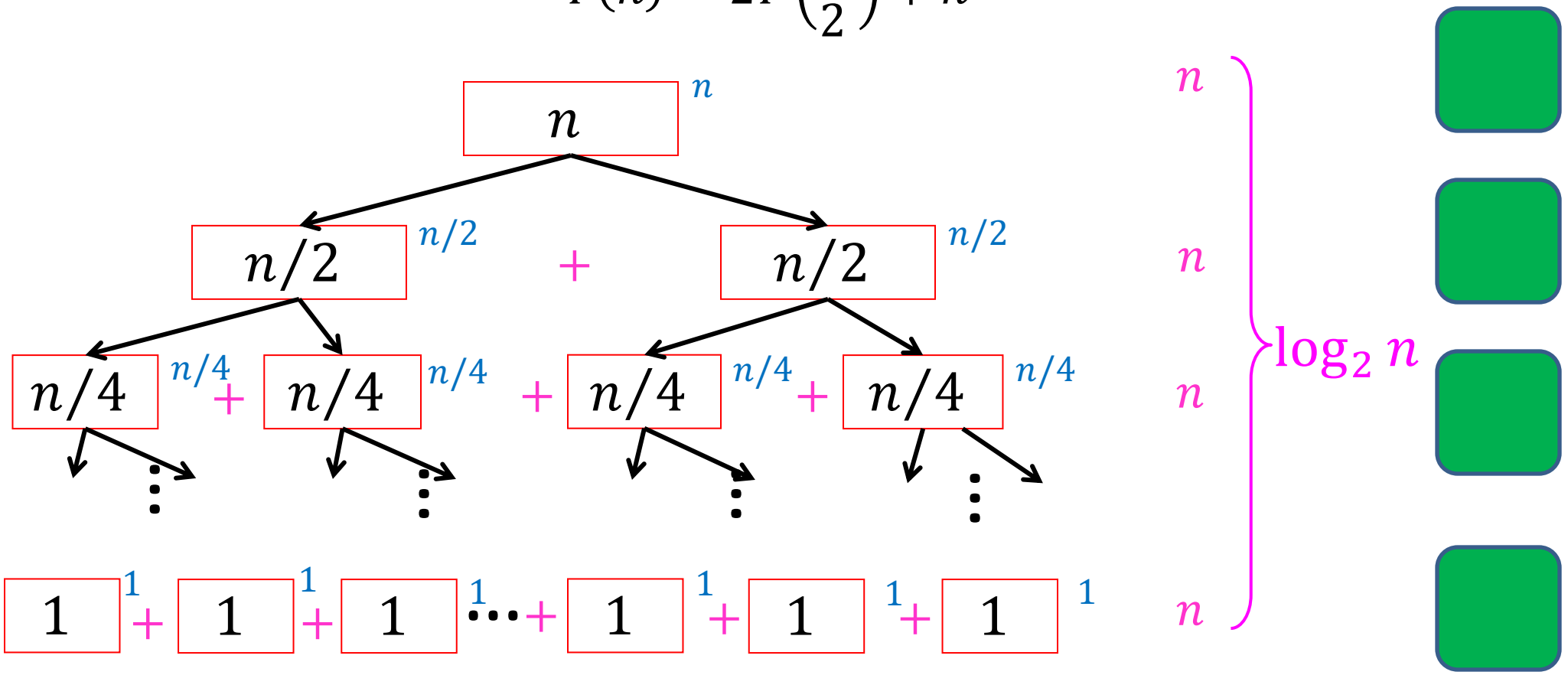
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

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

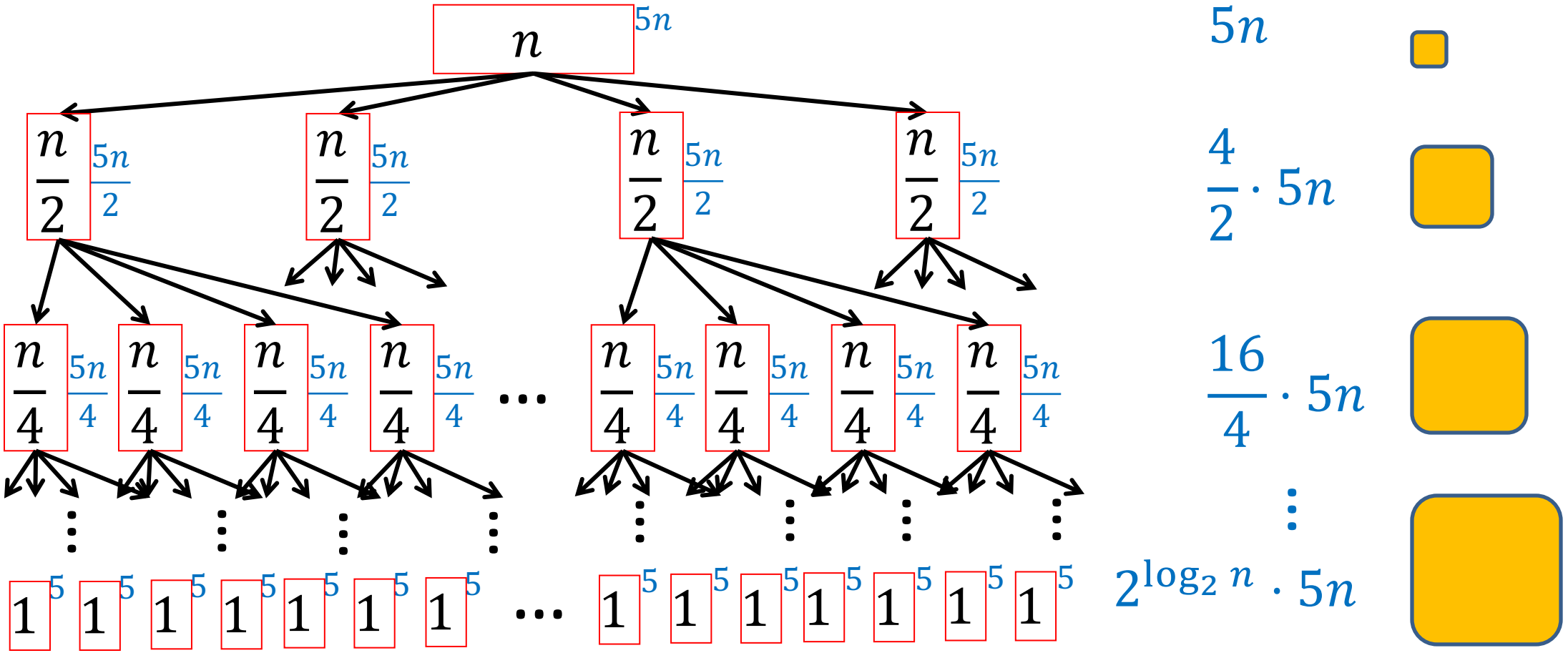
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Case 1

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

Tree method

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

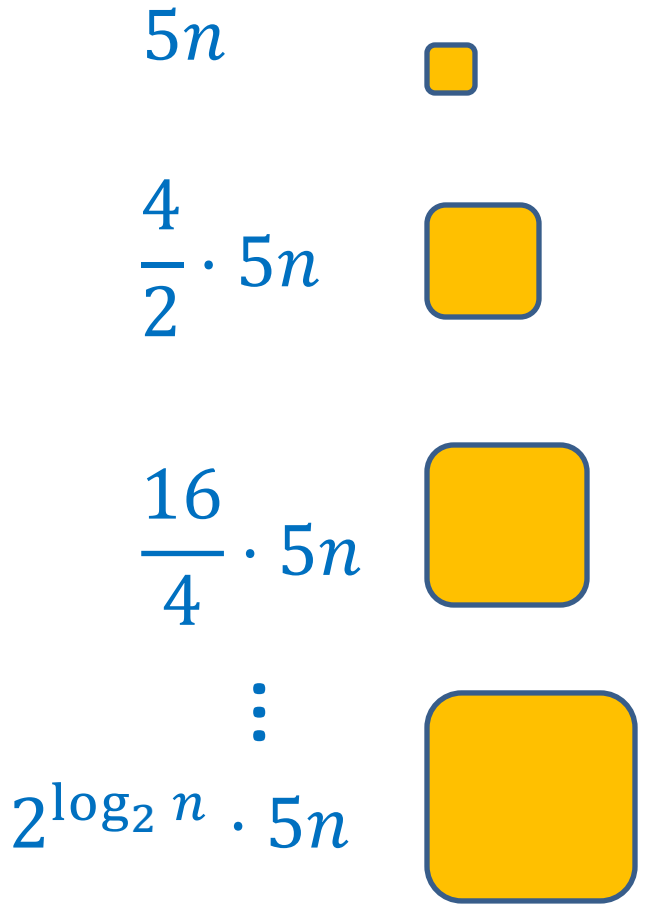


Tree method

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

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

Most of the work happening in the leaves



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

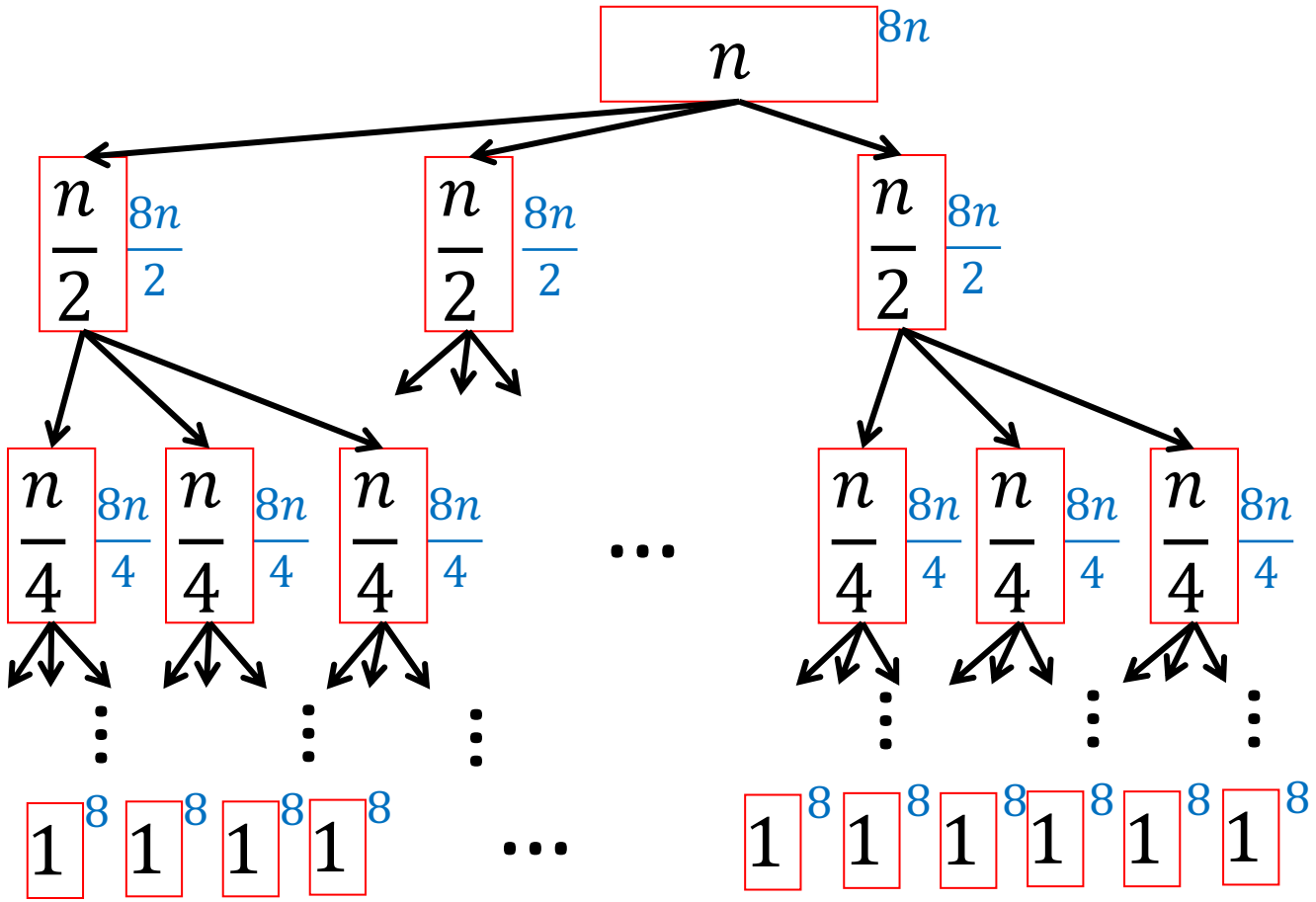
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Case 1

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

Karatsuba

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



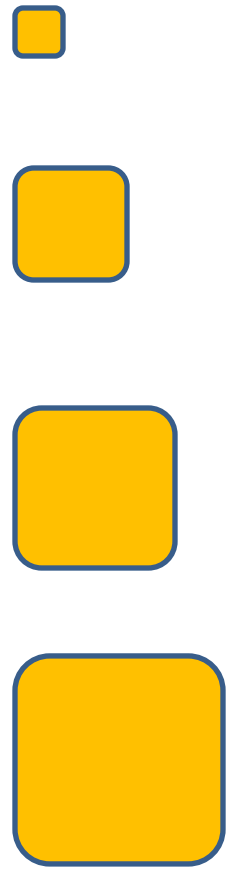
$$8 \cdot 1n$$

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

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

$$\vdots$$

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



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

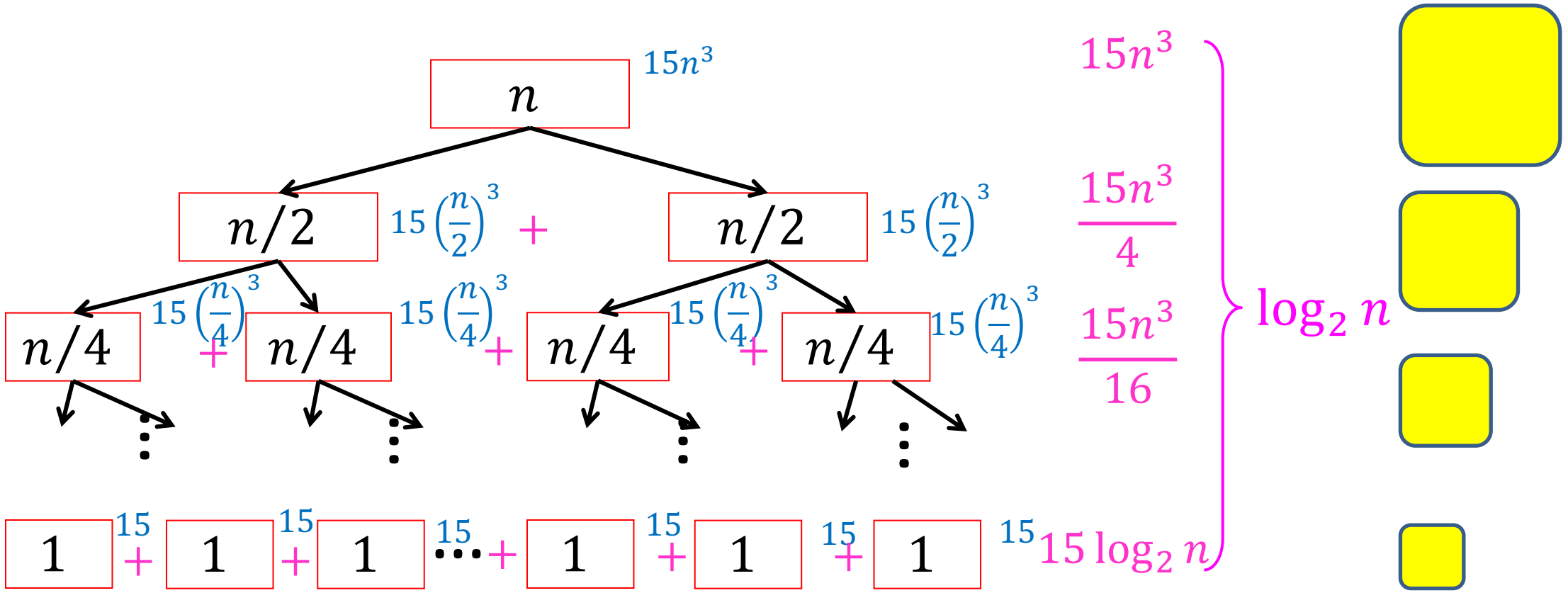
$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$

Case 3

$$\Theta(n^3)$$

Tree method

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

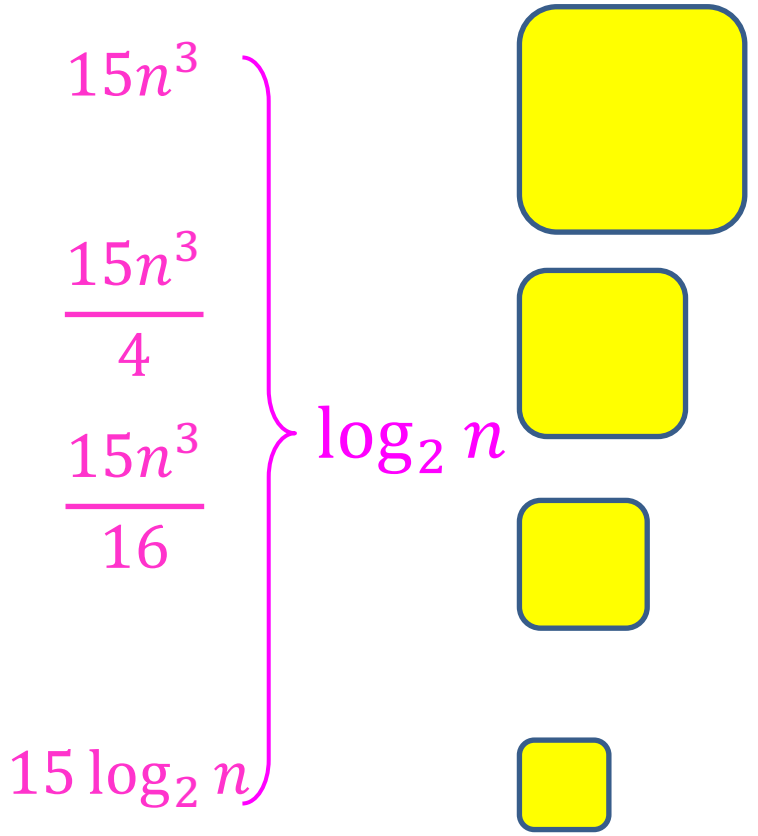


Tree method

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

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

Most of the work happening at the top



Recurrence Solving Techniques

Tree

Guess/Check

"Cookbook"

Substitution

Substitution Method

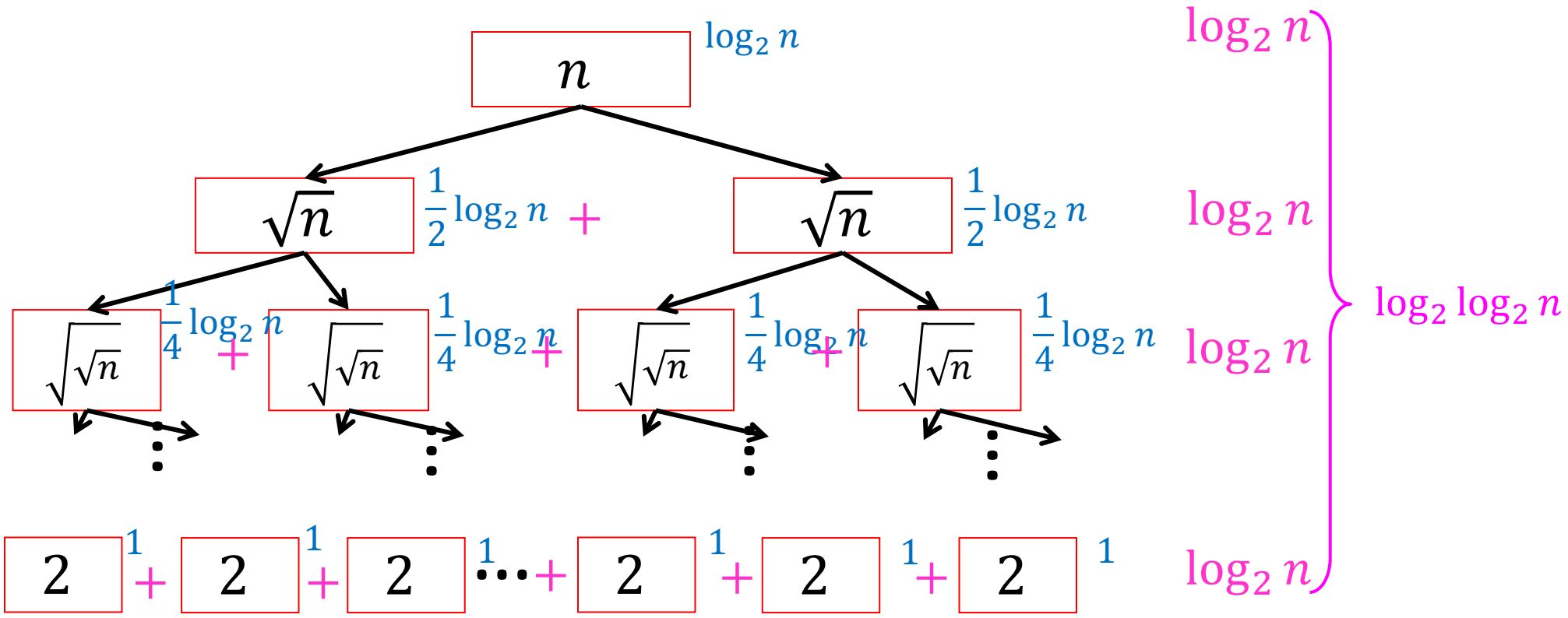
- Idea: take a “difficult” recurrence, re-express it such that one of our other methods applies.
- Example:

$$T(n) = 2T(\sqrt{n}) + \log_2 n$$

Tree method

$$\log_2 n^{1/2} = \frac{1}{2}\log_2 n$$

$$T(n) = 2T(\sqrt{n}) + \log_2 n$$



$$T(n) = O(\log_2 n \cdot \log_2 \log_2 n)$$

Substitution Method

$$T(n) = 2T(\sqrt{n}) + \log_2 n$$
$$= 2T\left(n^{1/2}\right) + \log_2 n$$

I don't like the ½ in the exponent

Let $n = 2^m$, i.e. $m = \log_2 n$

Now the variable is in the exponent on both sides!

$T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$ Rewrite in terms of exponent!

Let $S(m) = 2S\left(\frac{m}{2}\right) + m$ Case 2!

Let $S(m) = \Theta(m \log m)$ Substitute Back

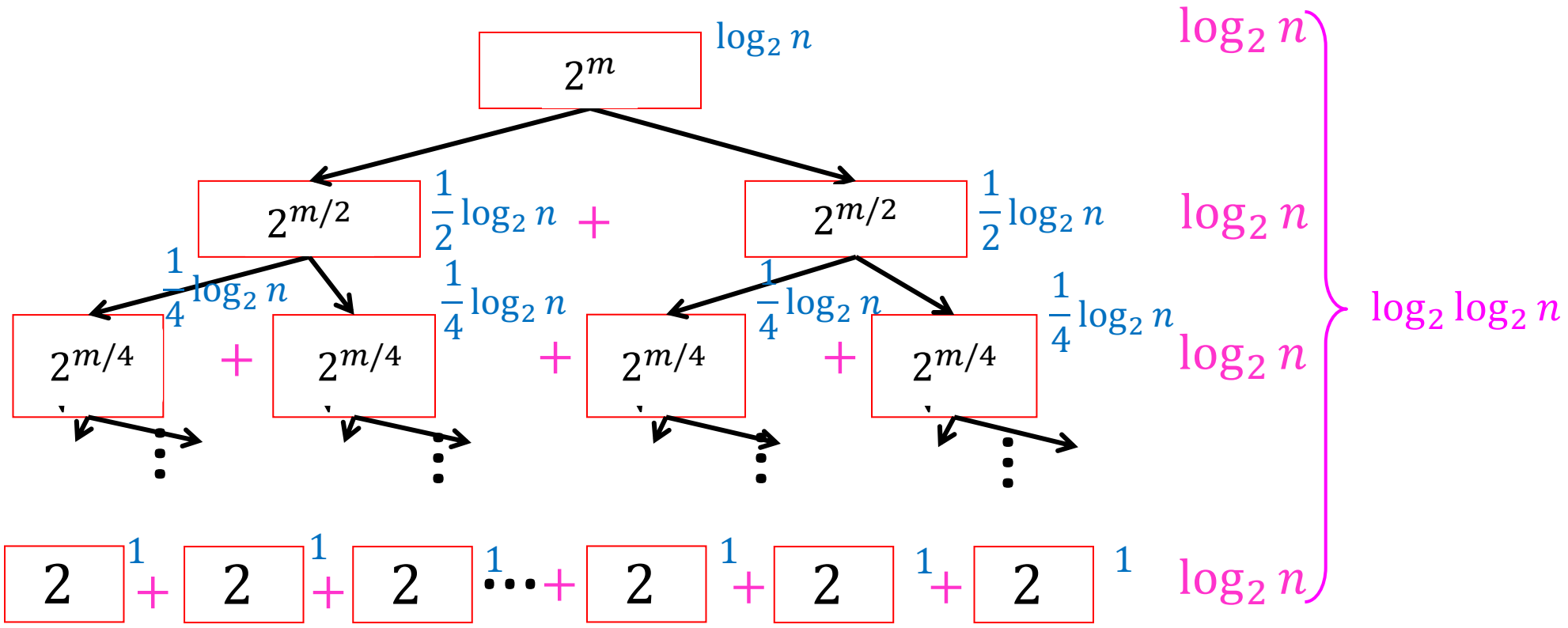
S will operate exactly as T, just redefined in terms of the exponent

$S(m) = T(2^m)$

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

Tree method

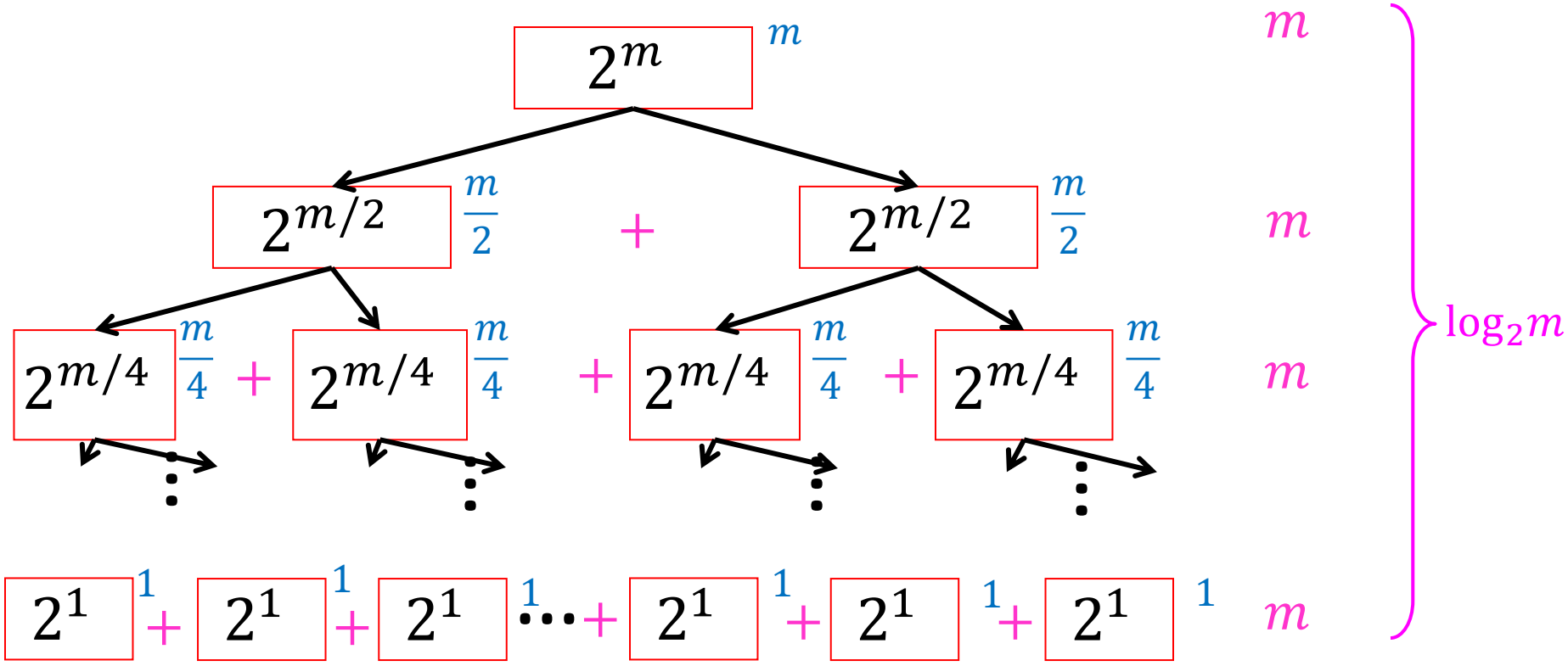
$$n = 2^m \qquad T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$$



Tree method

$n = 2^m$

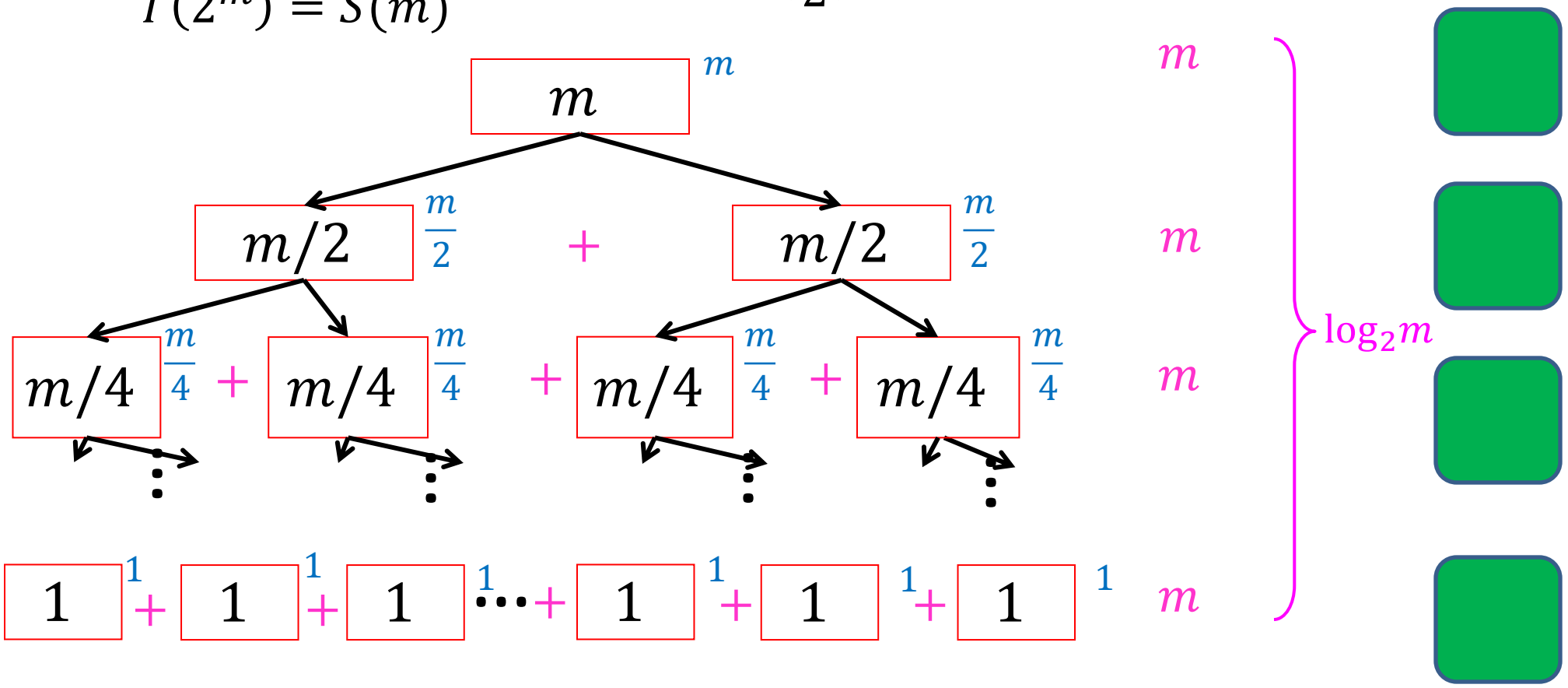
$T(2^m) = 2T(2^{m/2}) + m$



Tree method

$$n = 2^m$$
$$T(2^m) = S(m)$$

$$S(m) = 2S\left(\frac{m}{2}\right) + m$$



$$T(n) = O(m \cdot \log_2 m) = O(\log_2 n \cdot \log_2 \log_2 n)$$