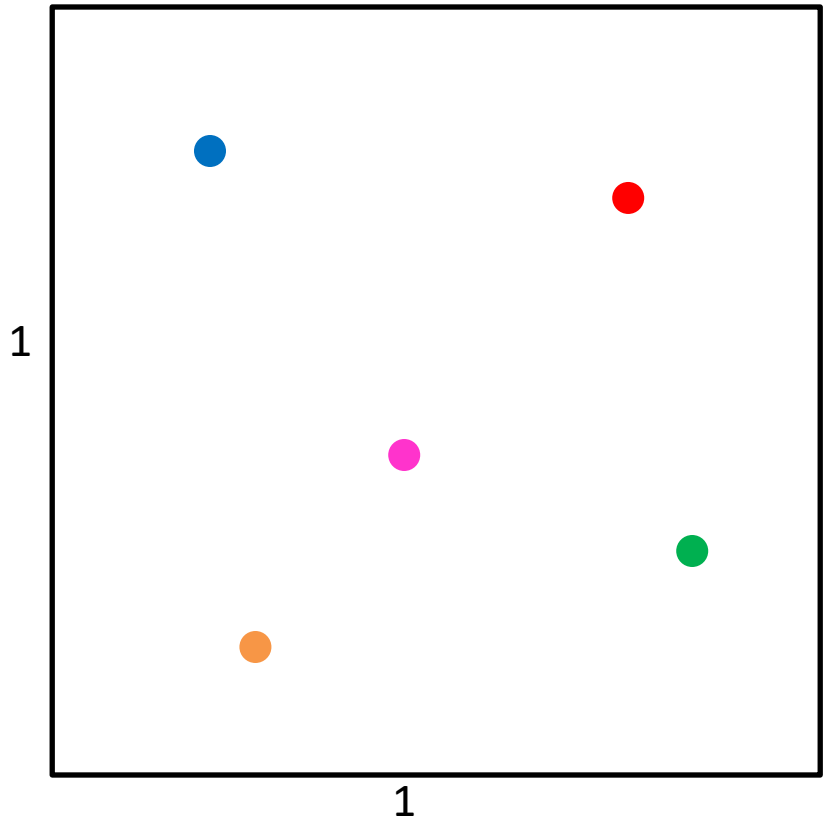
CS4102 Algorithms

Fall 2019

Warm up

Given any 5 points on the unit square, show there's always a pair distance $\leq \frac{\sqrt{2}}{2}$ apart



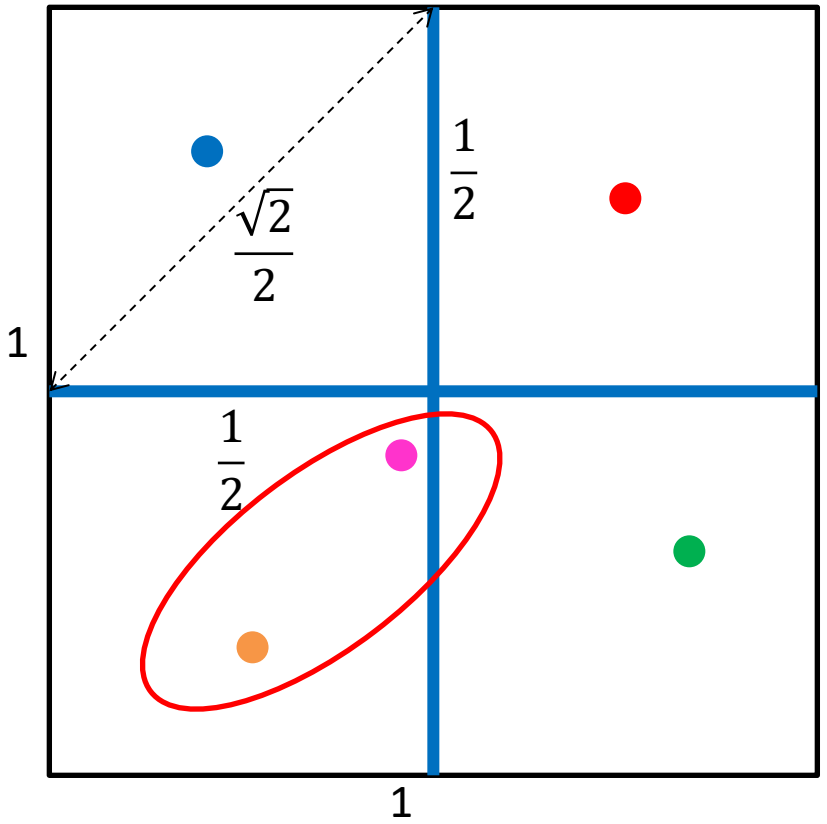
CS4102 Algorithms

Fall 2019

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

Given 5 points, two must share the same quadrant

Pigeonhole Principle!



Today's Keywords

- Solving recurrences
- Cookbook Method
- Master Theorem
- Substitution Method

CLRS Readings

- Chapter 4

Homeworks

Saturday, September 14 at 11pm

- Hw1 due ~~Thursday, September 12 at 11pm~~
  - Written (use Latex!) – Submit BOTH pdf and zip!
  - Asymptotic notation
  - Recurrences
  - Divide and Conquer

Recurrence Solving Techniques

Tree

Guess/Check (induction)

"Cookbook"

Substitution

Induction (review)

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

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

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

Technically, called *strong induction*

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 base case or to the inductive hypothesis

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

Karatsuba Guess and Check

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

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

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

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

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

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

$$= 3 \cdot 24 \frac{(x_0+1)^{\log_2 3}}{2^{\log_2 3}} - 16(x_0+1)$$

$$= 24(x_0+1)^{\log_2 3} - 16(x_0+1)$$

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 the form:

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

Remember…

- Better Attendance: $T(n) = T\left(\frac{n}{2}\right) + 2$
- 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

$a = b^{\log_b a}$

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

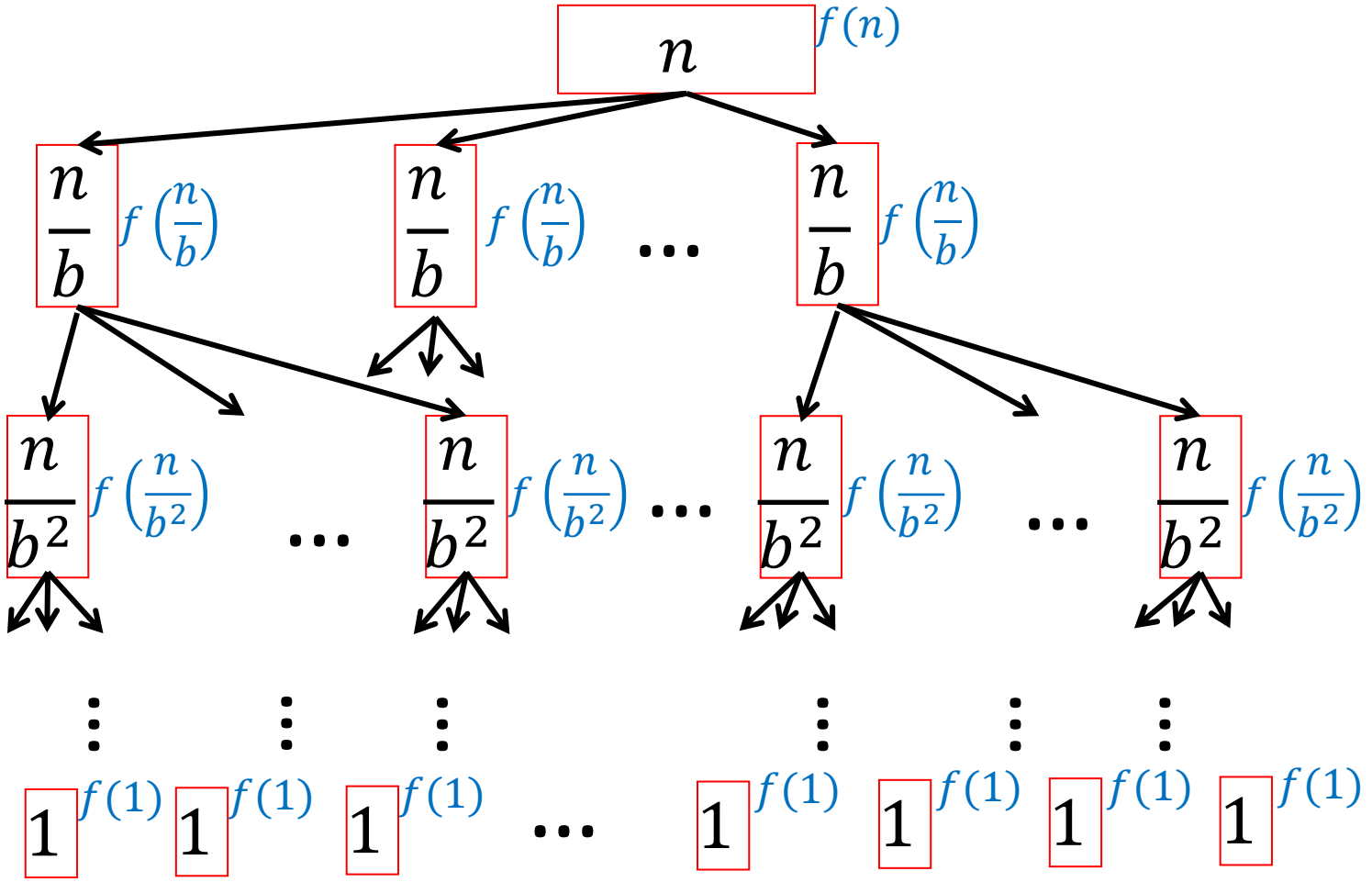
$$T(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$

$a^{\log_b n}$

$\left(b^{\log_b a}\right)^{\log_b n}$

$\left(b^{\log_b n}\right)^{\log_b a}$

$n^{\log_b a}$



$f(n)$

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

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

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

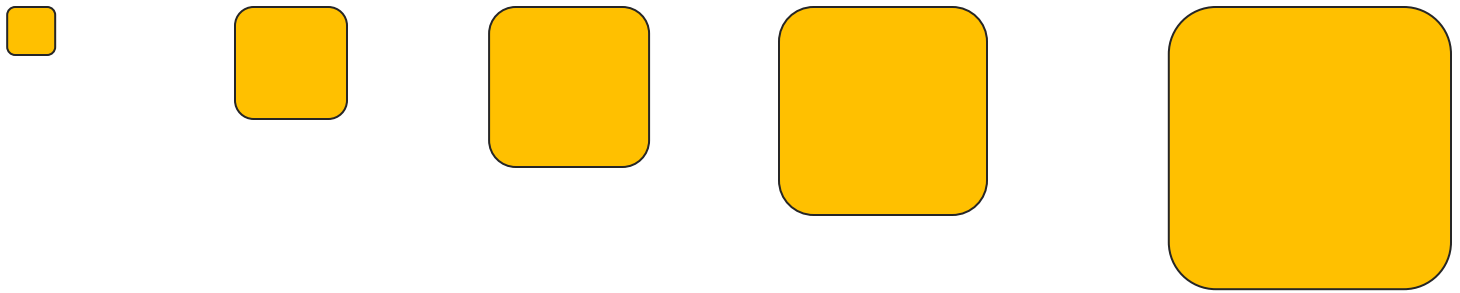
$n^{\log_b a} f(1)$

3 Cases

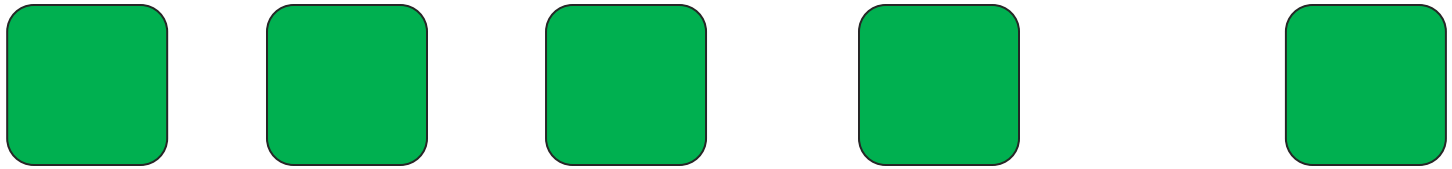
$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + a^3 f\left(\frac{n}{b^3}\right) + \cdots + a^L f\left(\frac{n}{b^L}\right)$$

$$L = \log_b n$$

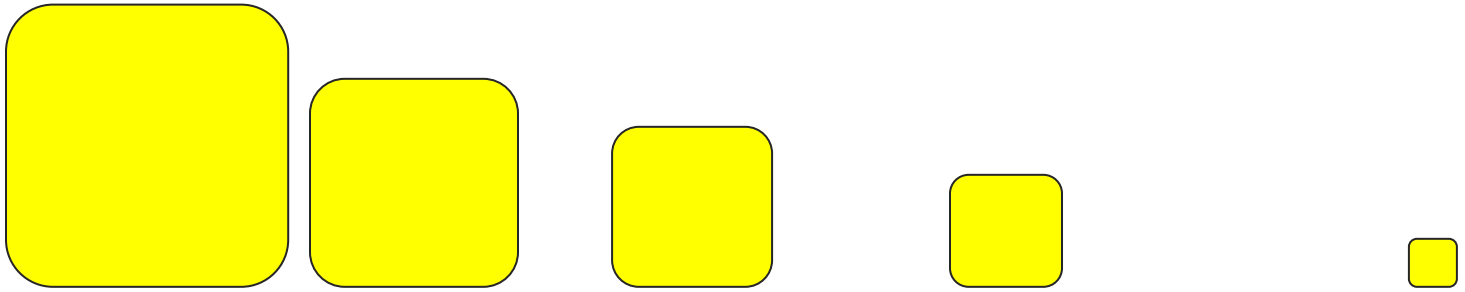
Case 1:
Most work happens at the leaves



Case 2:
Work happens consistently throughout



Case 3:
Most work happens at top of tree



Master Theorem

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

$\log_b n = \frac{\log_2 n}{\log_2 b}$

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

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

Let: $\exists c, n_0 > 0 : \forall n > n_0$

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

Insert math here…

Proof of Case 1

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

$$\leq c\left(n^{\log_b a - \varepsilon} + a\left(\frac{n}{b}\right)^{\log_b a - \varepsilon} + a^2\left(\frac{n}{b^2}\right)^{\log_b a - \varepsilon} + \cdots + a^{L-1}\left(\frac{n}{b^{L-1}}\right)^{\log_b a - \varepsilon}\right) + a^L f(1)$$

$$= c n^{\log_b a - \varepsilon}\left(1 + \frac{a}{b^{\log_b a - \varepsilon}} + \frac{a^2}{(b^2)^{\log_b a - \varepsilon}} + \cdots + \frac{a^{L-1}}{(b^{L-1})^{\log_b a - \varepsilon}}\right) + a^L f(1)$$

$$b^{\log_b a - \varepsilon} = a b^{-\varepsilon} \qquad ab^{-\varepsilon} \qquad a^2 b^{-2\varepsilon} \qquad a^{L-1} \cdot b^{-\varepsilon(L-1)}$$

$$= c n^{\log_b a - \varepsilon}\left(1 + b^{\varepsilon} + b^{2\varepsilon} + \cdots + b^{\varepsilon(L-1)}\right) + a^L f(1)$$

$$= c n^{\log_b a - \varepsilon}\left(\frac{b^{\varepsilon L} - 1}{b^{\varepsilon} - 1}\right) + a^L f(1)$$

Proof of Case 1

$$T(n) \le c\, n^{\log_b a - \varepsilon} \left(\frac{b^{\varepsilon(\log_b n)} - 1}{b^{\varepsilon} - 1} \right) + a^{\log_b n} f(1)$$

$$= c\, n^{\log_b a - \varepsilon} \left(n^{\varepsilon} - 1 \right) \left(\frac{1}{b^{\varepsilon} - 1} \right) + n^{\log_b a} c_2$$

$$= c_4 n^{\log_b a} - c_4 n^{\log_b a - \varepsilon} + n^{\log_b a} c_2 \le c_5 n^{\log_b a}$$

$$\Rightarrow T(n) \in O\left(n^{\log_b a}\right)$$

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

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

$a = 2$
$b = 2$
$f(n) = n$

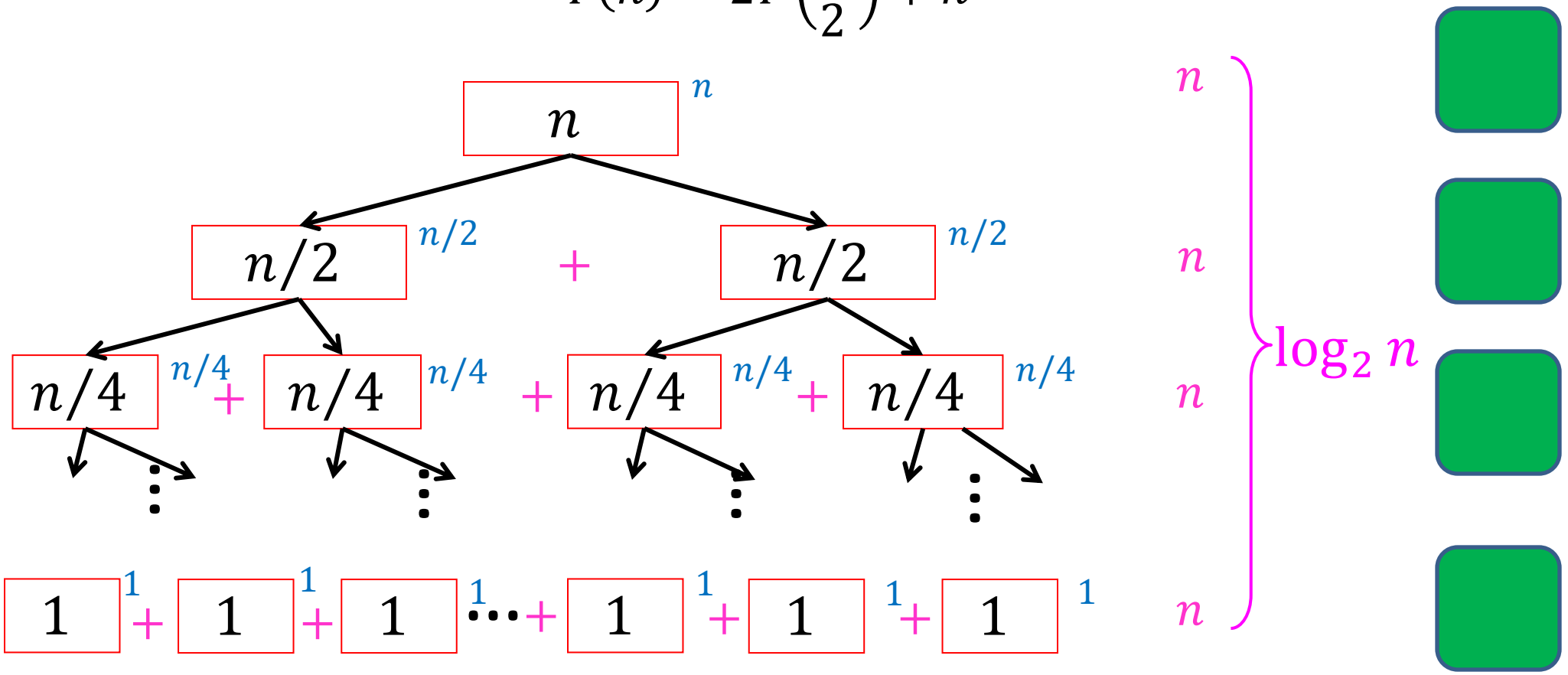
n ? $f(n) = n$

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

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

$a = 4$
$b = 2$
$f(n) = 5n$

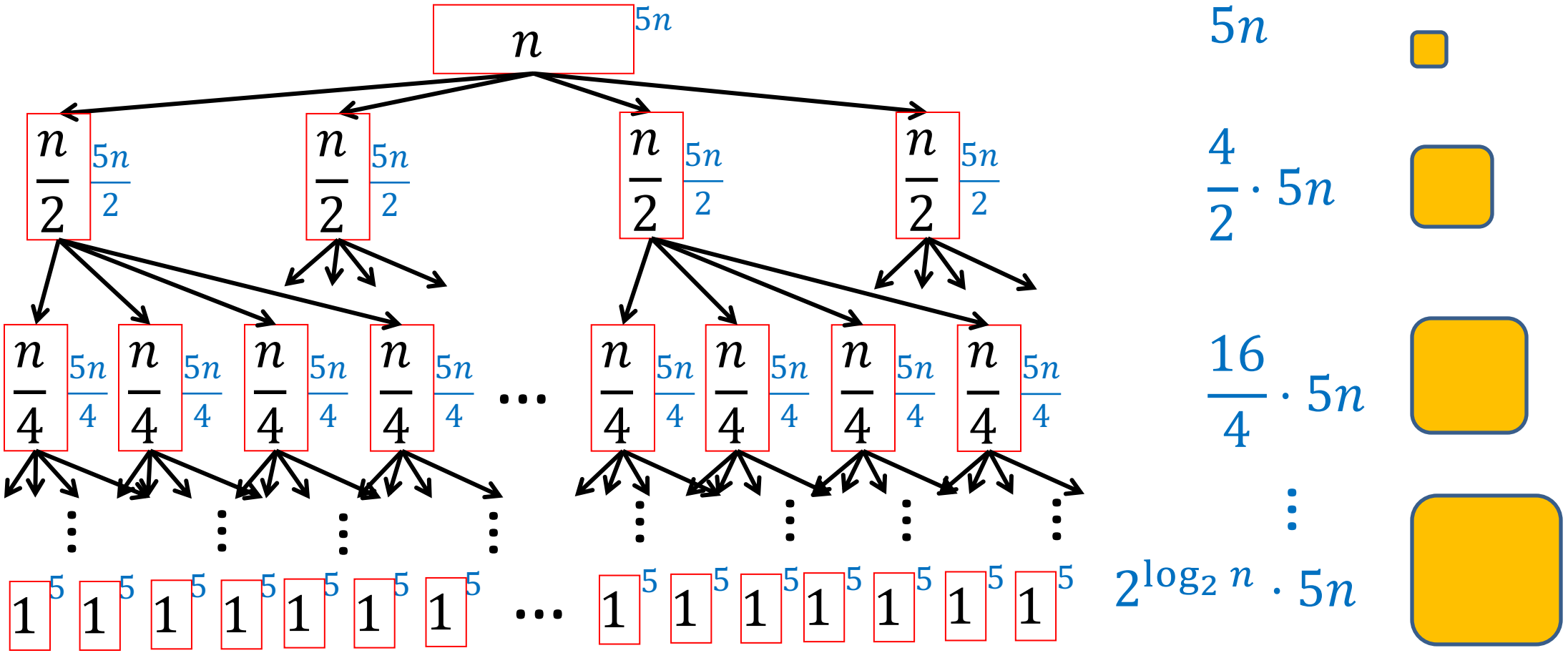
$f(n) ? n^2$
$\varepsilon = 1$
$f(n) \in O(n^{2-1})$

Case 1

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

Tree method

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

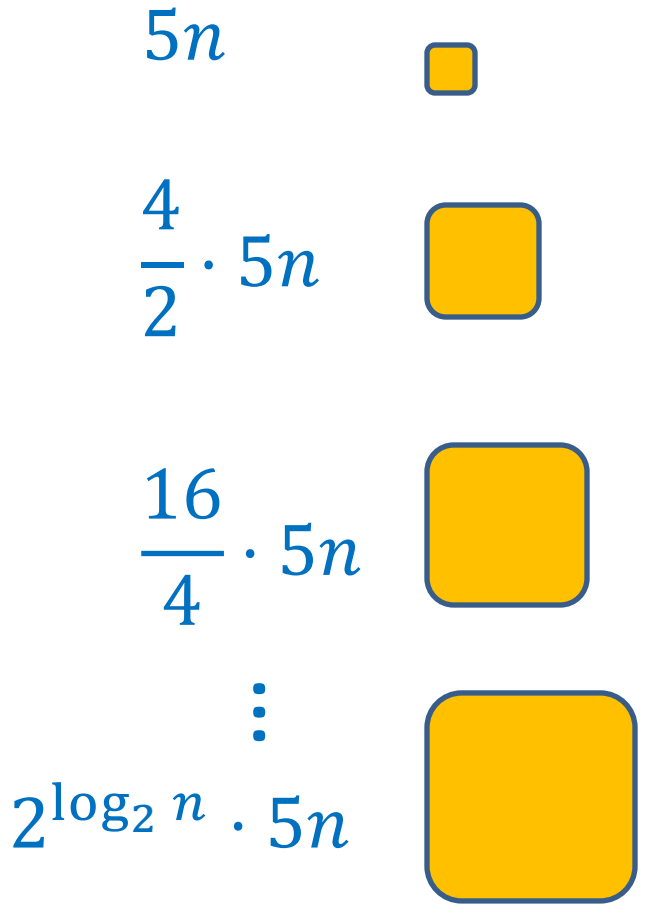


Tree method

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

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

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

$a=3$
$b=2$
$f(n)=8n$

$f(n)$ $n^{\log_2 3}$

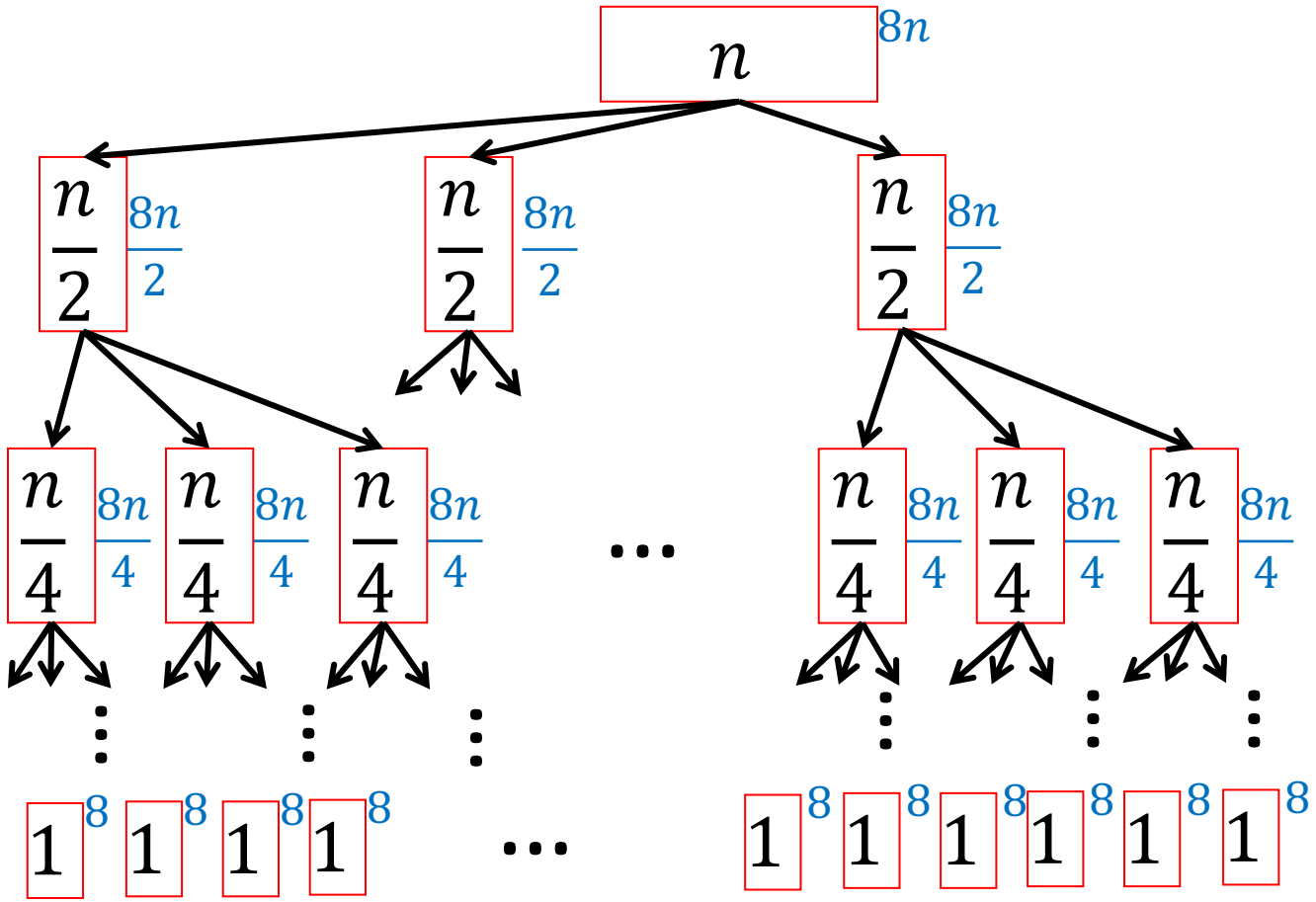
$8n \in O(n^{1.585})$

Case 1

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

Karatsuba

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



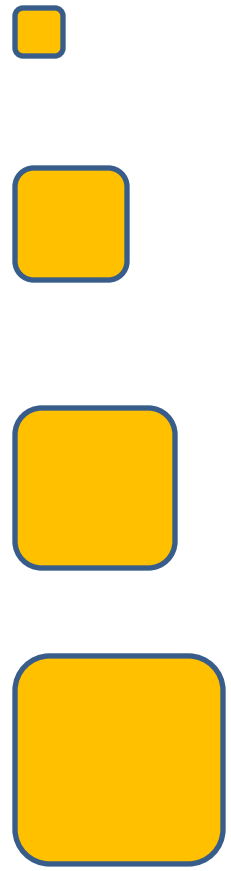
$$8 \cdot 1n$$

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

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

$$\vdots$$

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



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

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

$a = 2$
$b = 2$
$f(n) = 15n^3$

$15n^3 \in \Omega(n)$

$2 \cdot 15\left(\frac{n}{2}\right)^3 \leq c\,15n^3$

$\frac{15}{4}n^3 \leq c\,15n^3$

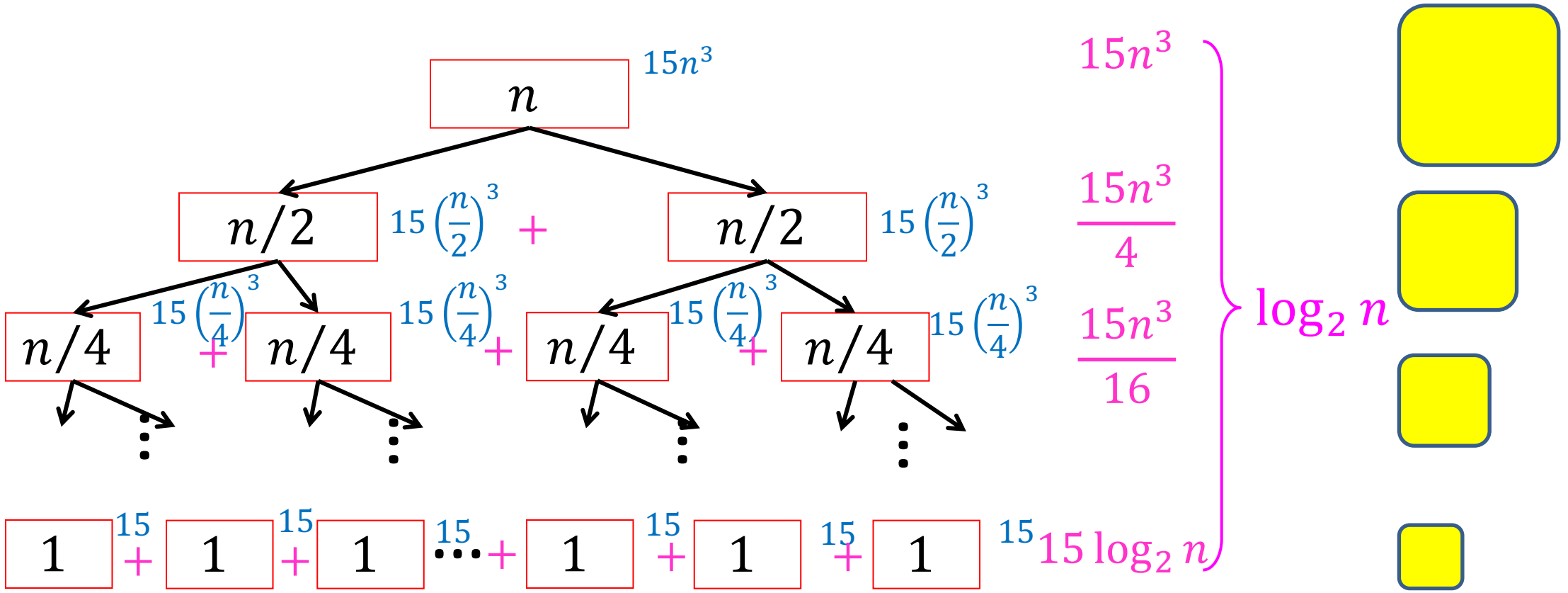
$c = \frac{1}{4}$

Case 3

$$\Theta(n^3)$$

Tree method

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

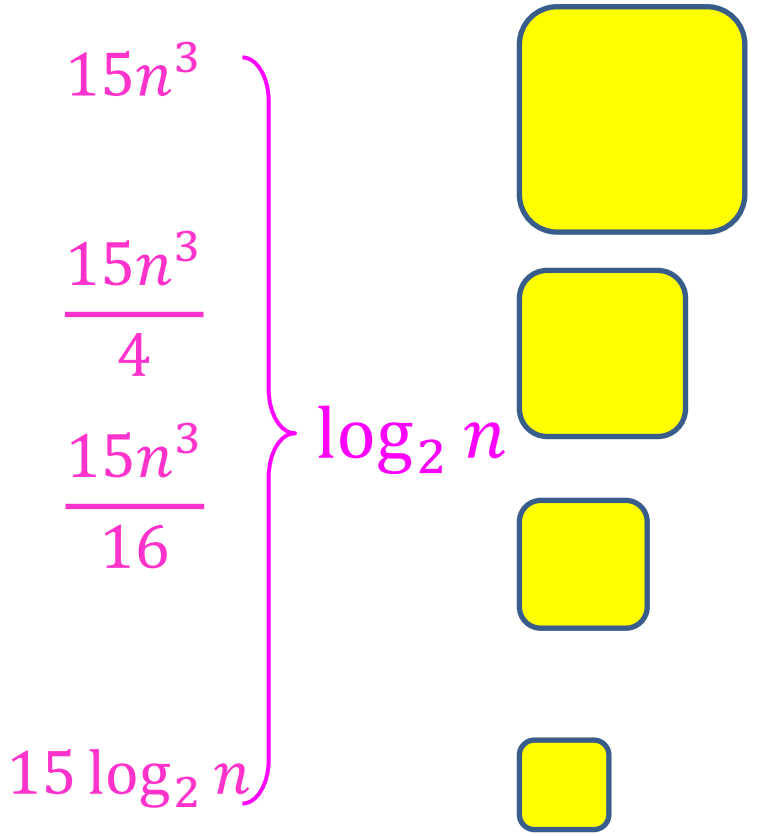


Tree method

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

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

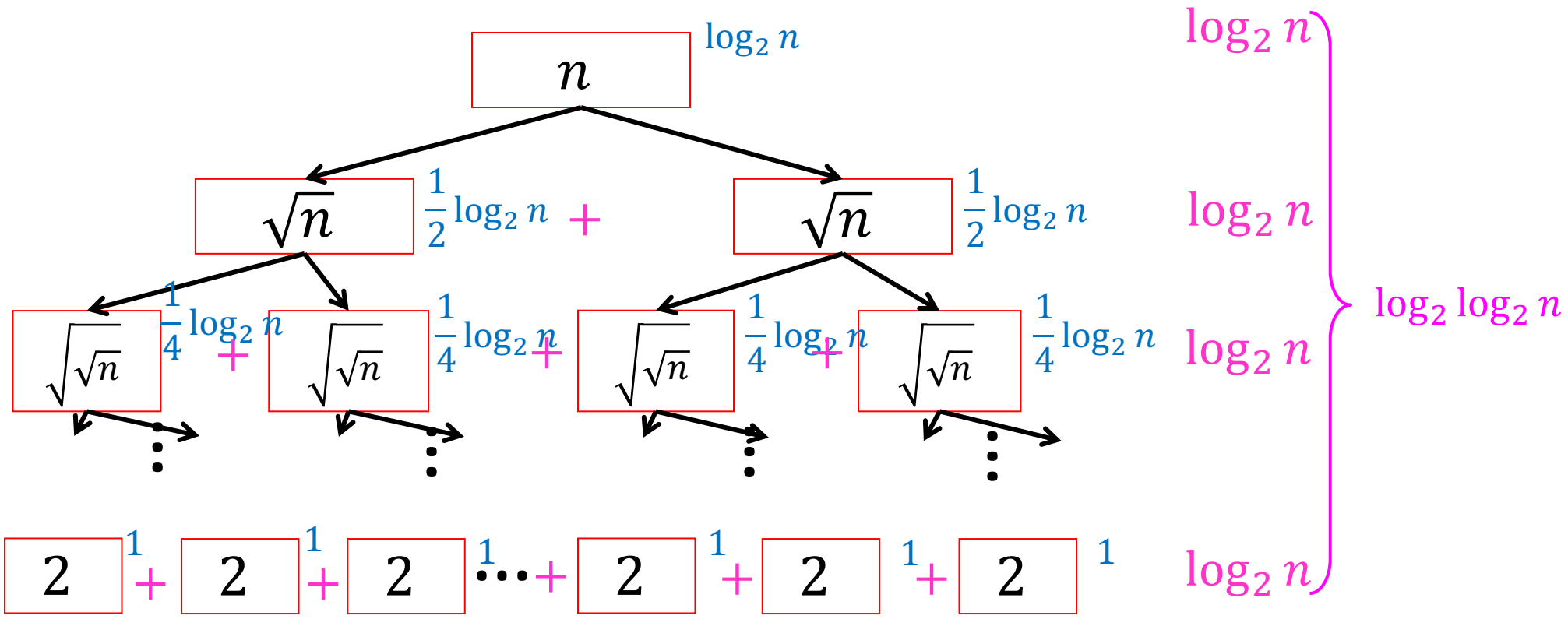
- Idea: take a “difficult” recurrence, re-express it such that one of our other methods applies.
- Example:

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

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

$a = 2$
$b = 2$
$f(m) = m$? $m^{\log_2 2} = m$

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!

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

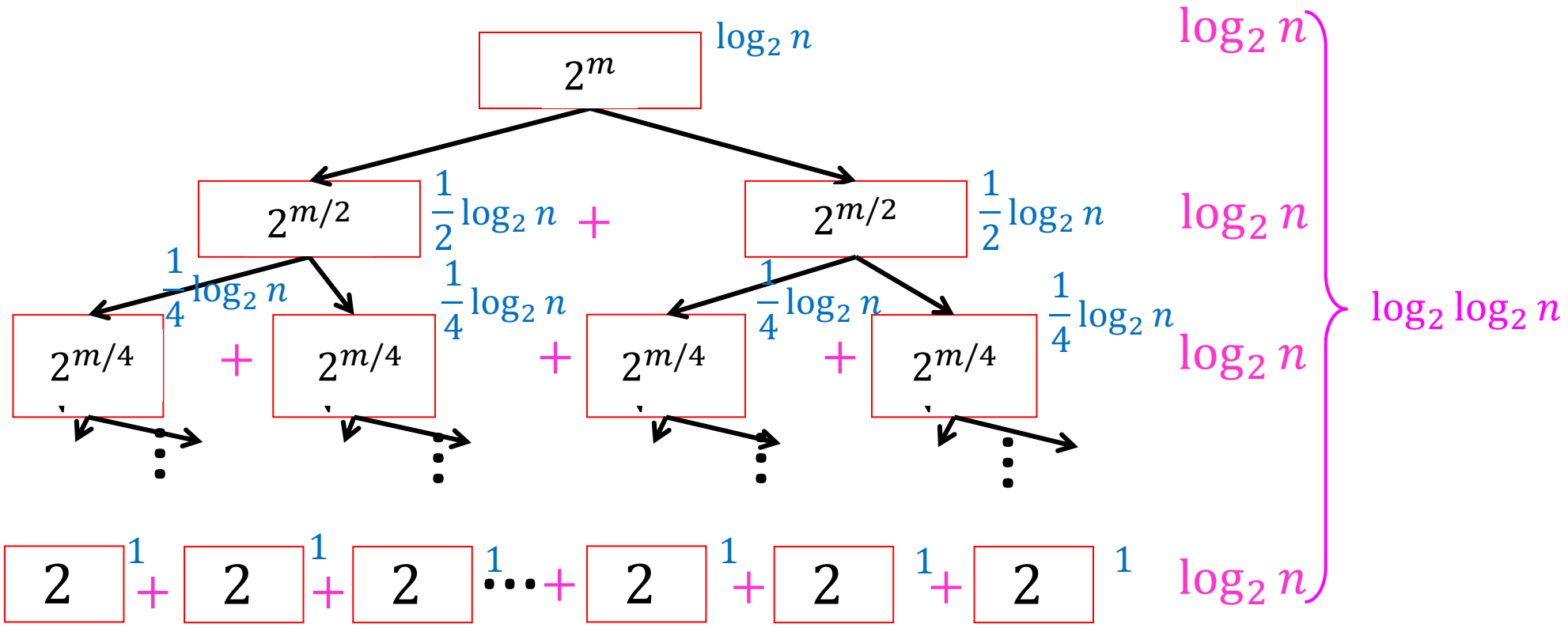
$S(m) = T(2^m)$

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

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

Tree method

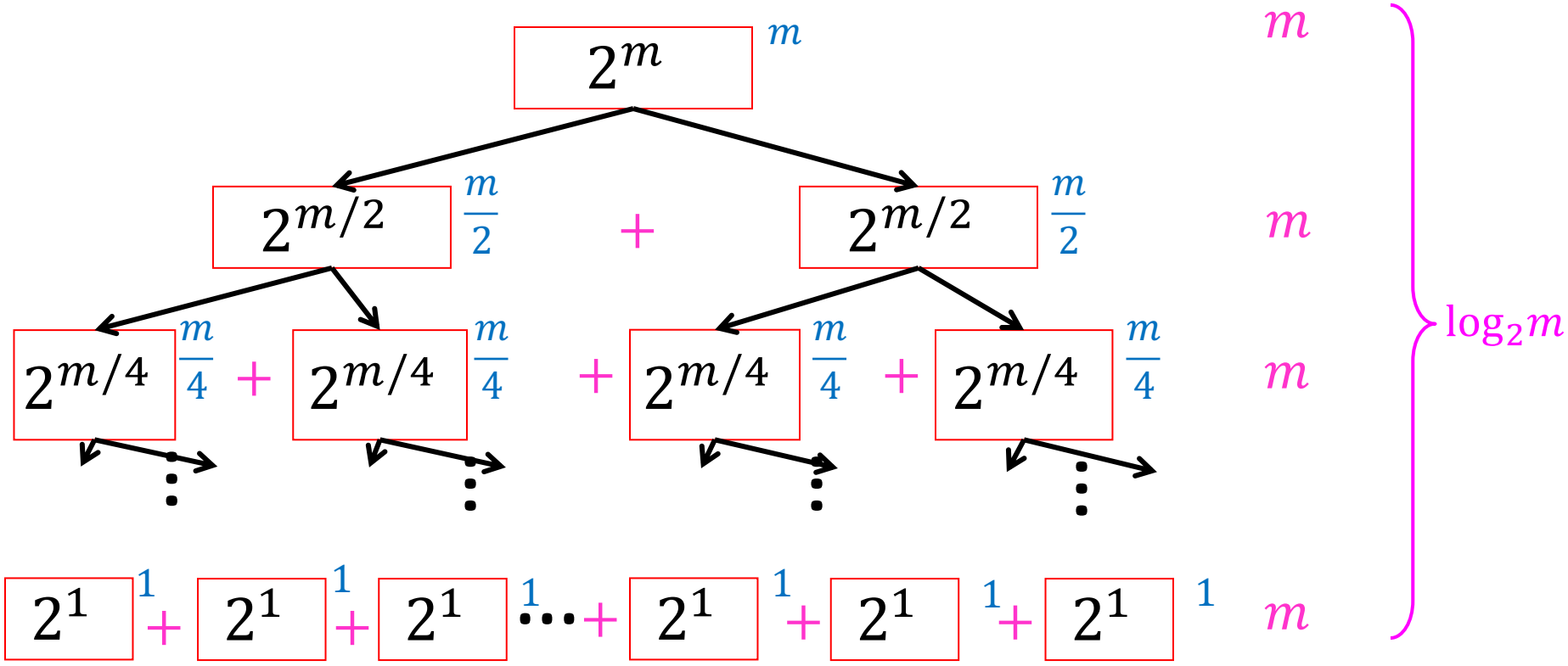
$$n = 2^m \qquad T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$$



Tree method

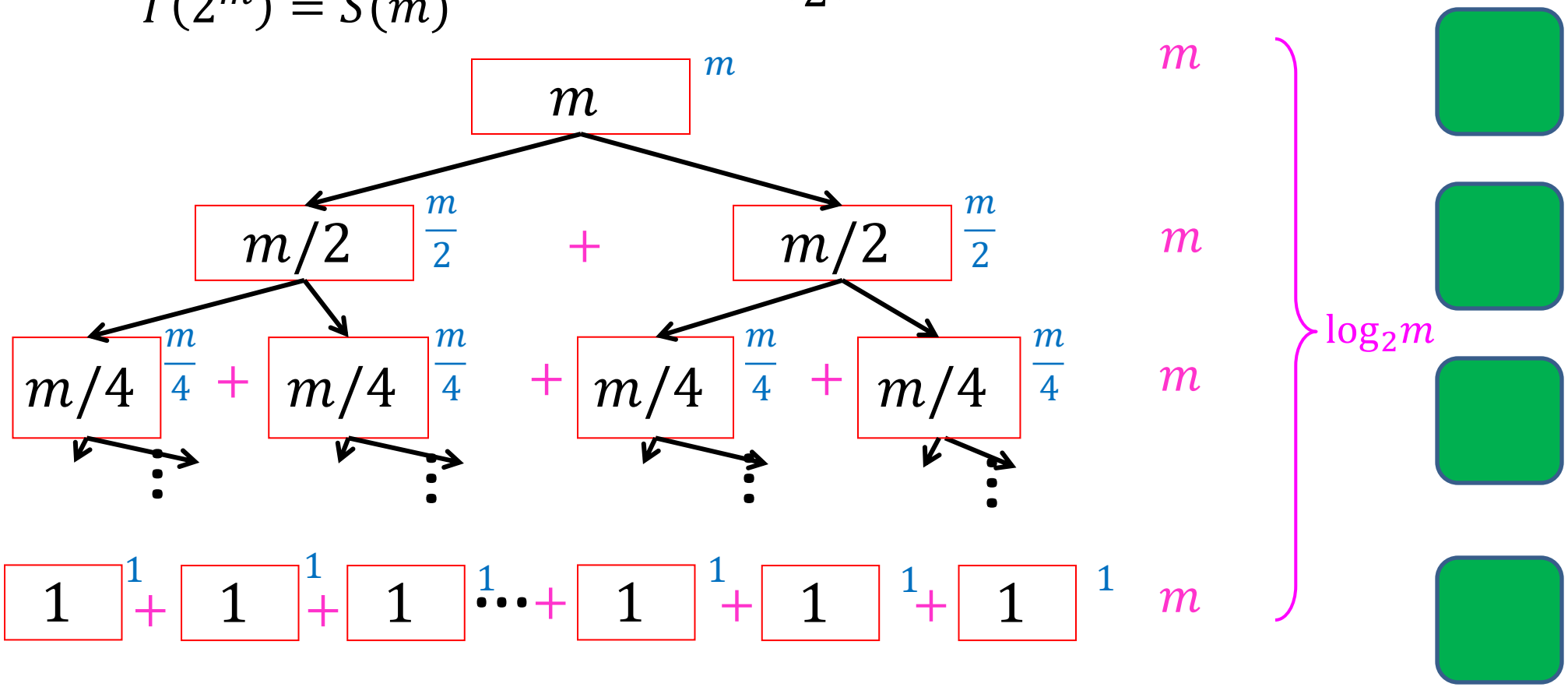
$n = 2^m$

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



Tree method

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



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