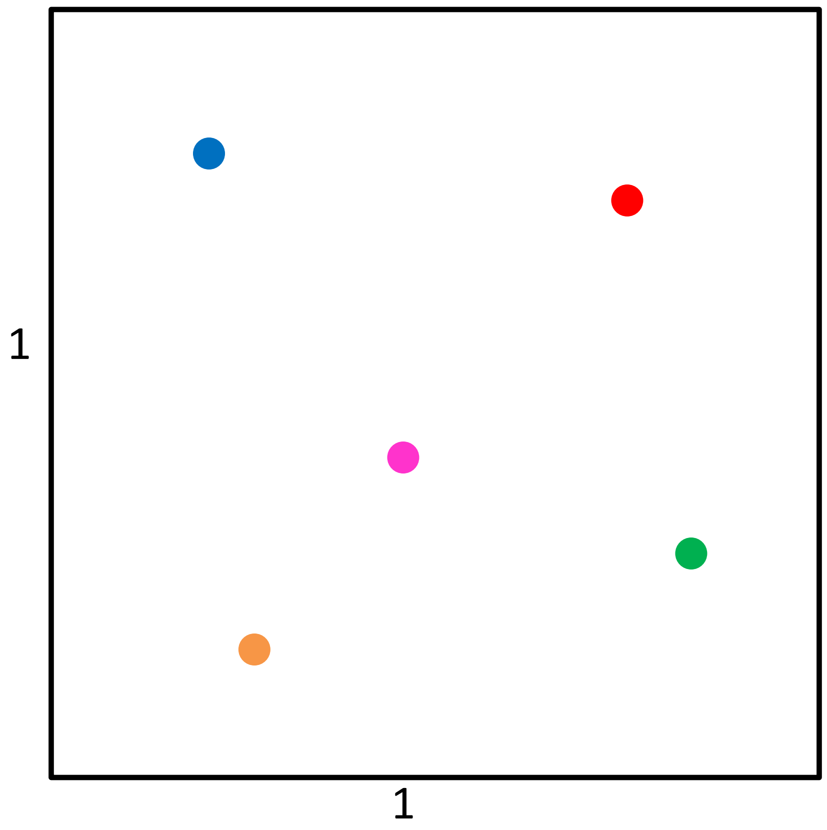
CS4102 Algorithms
Spring 2019

Warm up

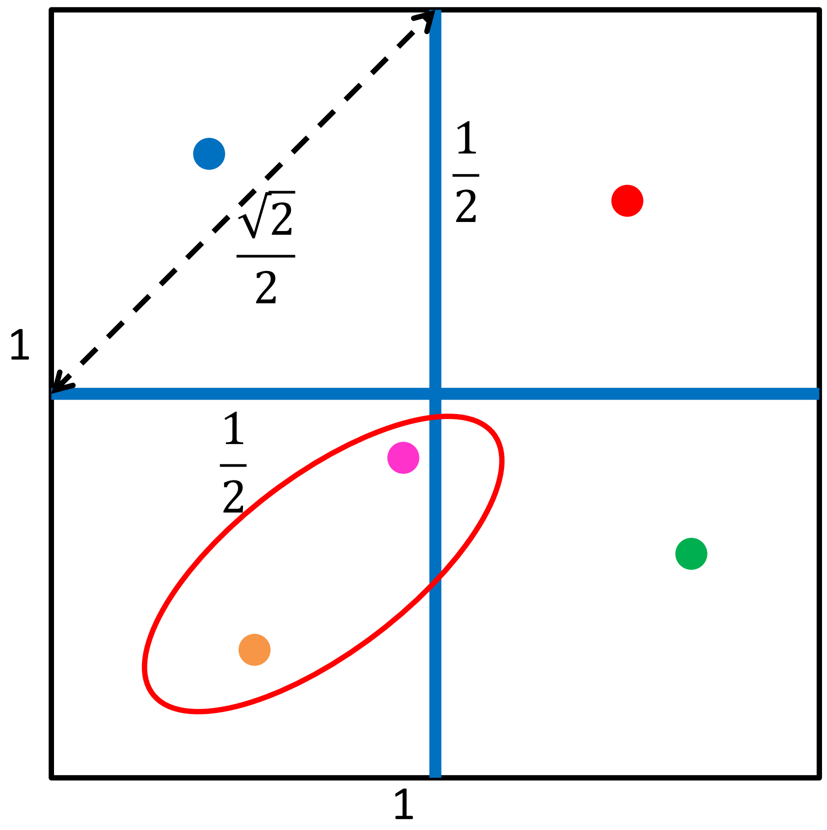
Given any 5 points on the unit square, show there's always a pair distance $\leq \frac{\sqrt{2}}{2}$ apart



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

- Hw1 due Sunday, Feb 3 at 11pm
  – Start early!
  – Written (use Latex!) – Submit BOTH pdf and zip!
  – Asymptotic notation
  – Recurrences
  – Divide and Conquer
- Hw2 released Monday, Feb 4 after class
  – Programming assignment (Python or Java)
  – Divide and conquer

Recurrence Solving Techniques

Tree

?  Guess/Check

(induction)

"Cookbook"

Substitution

Guess and Check Intuition

- To Prove: $T(n) = O(g(n))$
- Consider: $g_*(n) = O(g(n))$ *pick some specific function in $O(g(n))$*
- Goal: show $\exists n_0$ s.t. $\forall n > n_0, T(n) \leq g_*(n)$
  - (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
  - Hypothesis:
    - $\forall n \leq x_0, T(n) \leq g_*(n)$
  - Inductive step:
    - $T(x_0 + 1) \leq g_*(x_0 + 1)$

Recurrence Solving Techniques

Tree

Guess/Check

“Cookbook”

Substitution

Observation

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

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

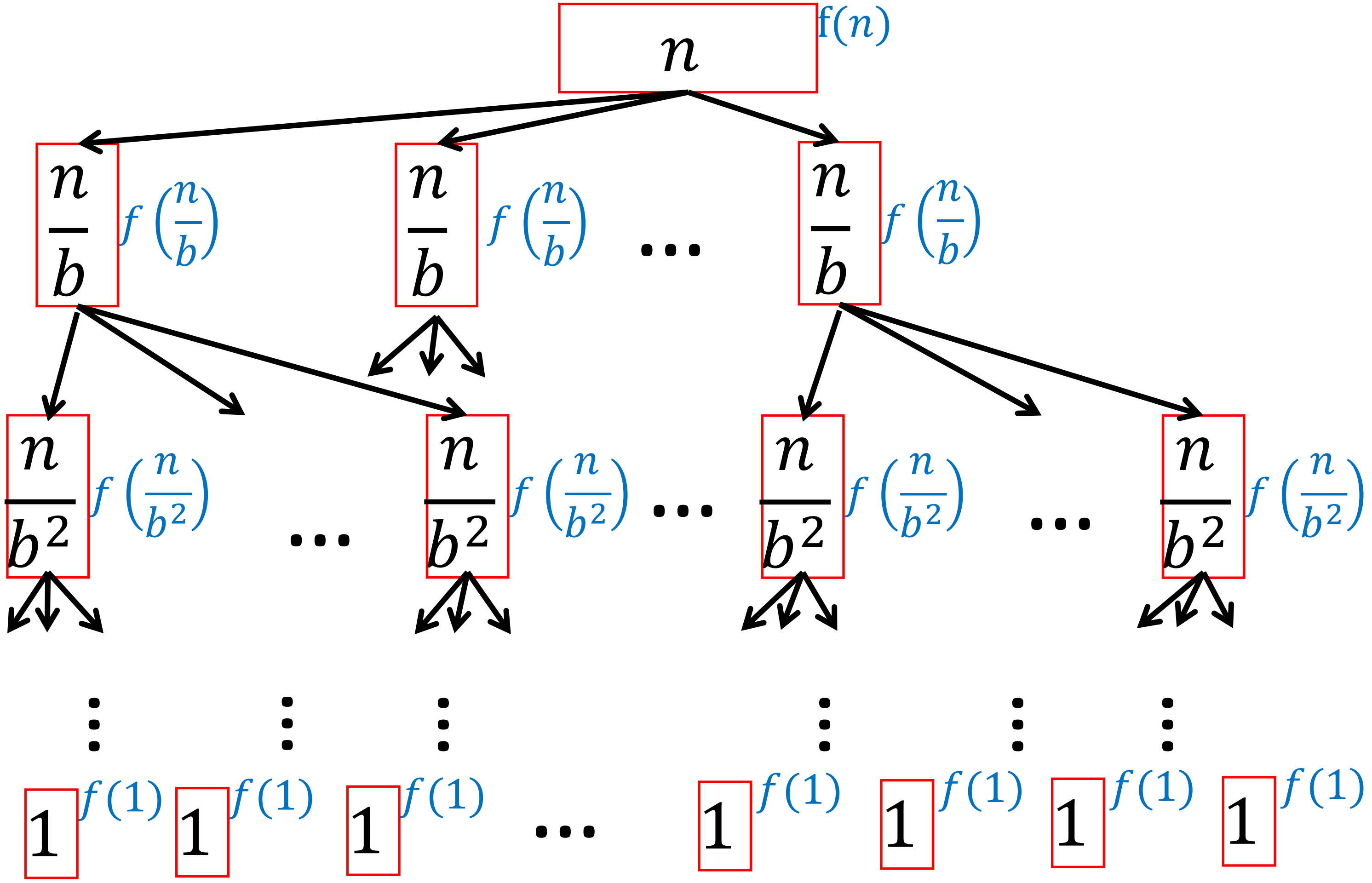
Remember…

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

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

General

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



$f(n)$

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

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

$\vdots$

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

Mathematical aside

$n^{2^2} = n^4$

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

$$a = b^{\log_b a} \qquad n = b^{\log_b n}$$

3 Cases

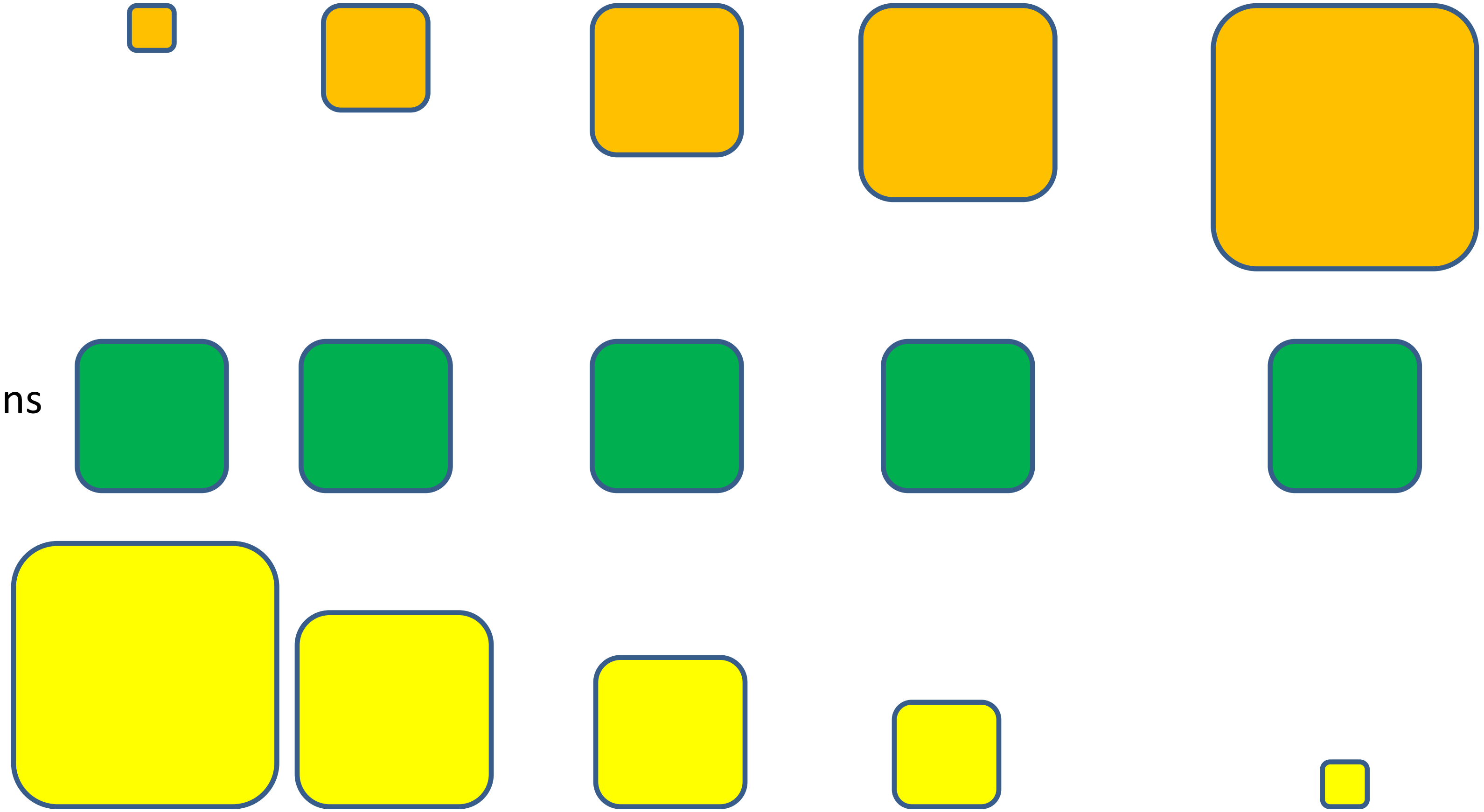
$L = \log_b n$

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

Case 1:
Most work happens at the leaves

Case 2:
Work happens consistently throughout

Case 3:
Most work happens at top of tree



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

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

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

Insert math here…

$$T(n) = O\left(n^{\log_b a}\right), \quad \text{Let } L = \log_b n$$

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

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

Proof of Case 1

$$b^{\log_b a - \varepsilon} = a \cdot b^{-\varepsilon}$$

$$b^{2(\log_b a - \varepsilon)} = a^2 \cdot b^{-2\varepsilon}$$

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

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

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

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

$$b^{\varepsilon \log_b n} = n^{\varepsilon}$$

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

Proof of Case 1

$$= c\, n^{\log_b a - \varepsilon}\left((n^{\varepsilon}-1)\cdot \underbrace{\cancel{\frac{1}{(b^{\varepsilon}-1)}}}_{c_2}\right) + \underbrace{\cancel{a^{\log_b n}}}_{n^{\log_b a}}\ \underbrace{\boxed{\cancel{\Theta(1)}}}_{c_3}$$

$$= c\, n^{\log_b a - \varepsilon}\left(n^{\varepsilon}-1\right) c_2 + c_3\, n^{\log_b a}$$

$c_4 = c \cdot c_2$

$$= c_4 n^{\log_b a} - c_4 n^{\log_b a - \varepsilon} + c_3 n^{\log_b a} = (c_3 + c_4) n^{\log_b a} - c_4 n^{\log_b a - \varepsilon}$$

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

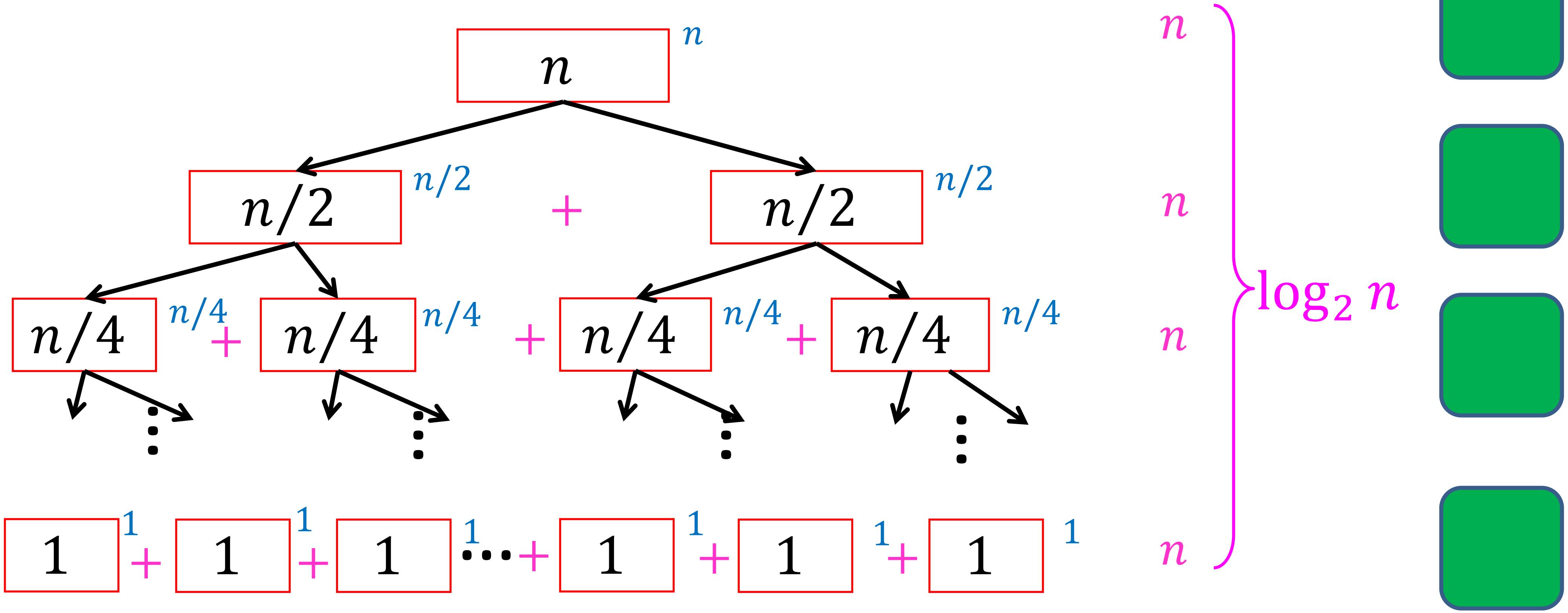
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

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

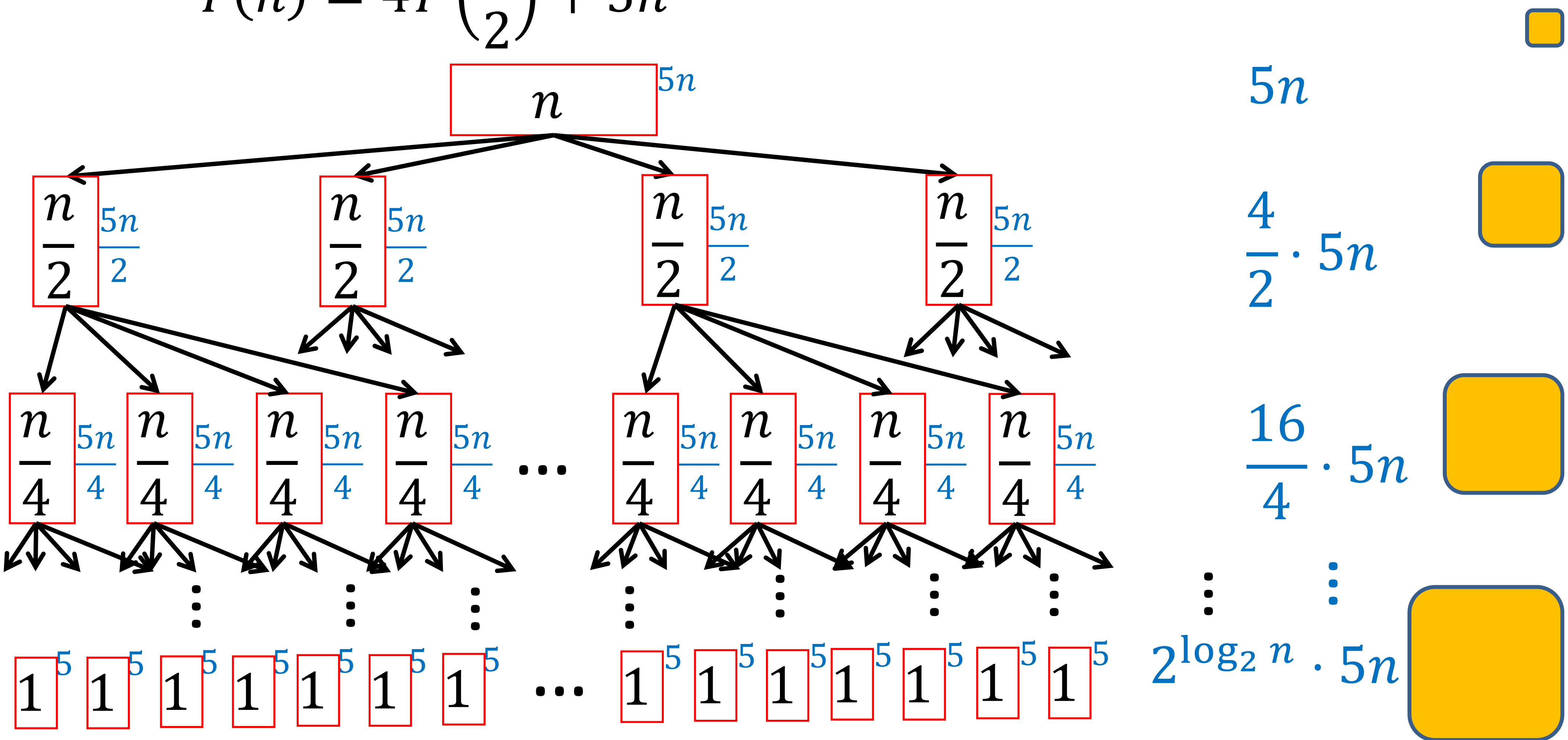
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Case 1

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

Tree method

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



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

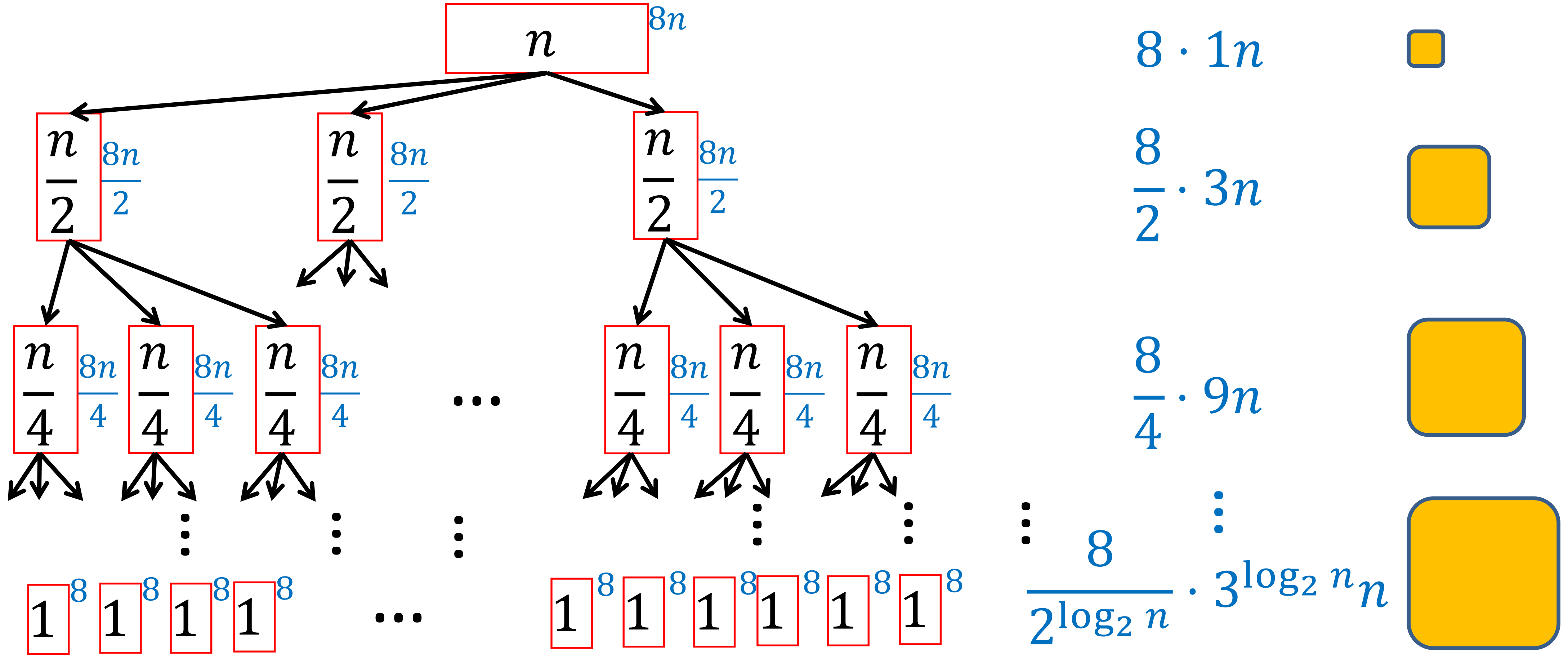
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

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 $\boxed{af\left(\frac{n}{b}\right) \leq cf(n)}$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

$2f\left(\frac{n}{2}\right) \leq cf(n)$

$\frac{2 \cdot 15 n^3}{8} \leq c\, 15n^3$

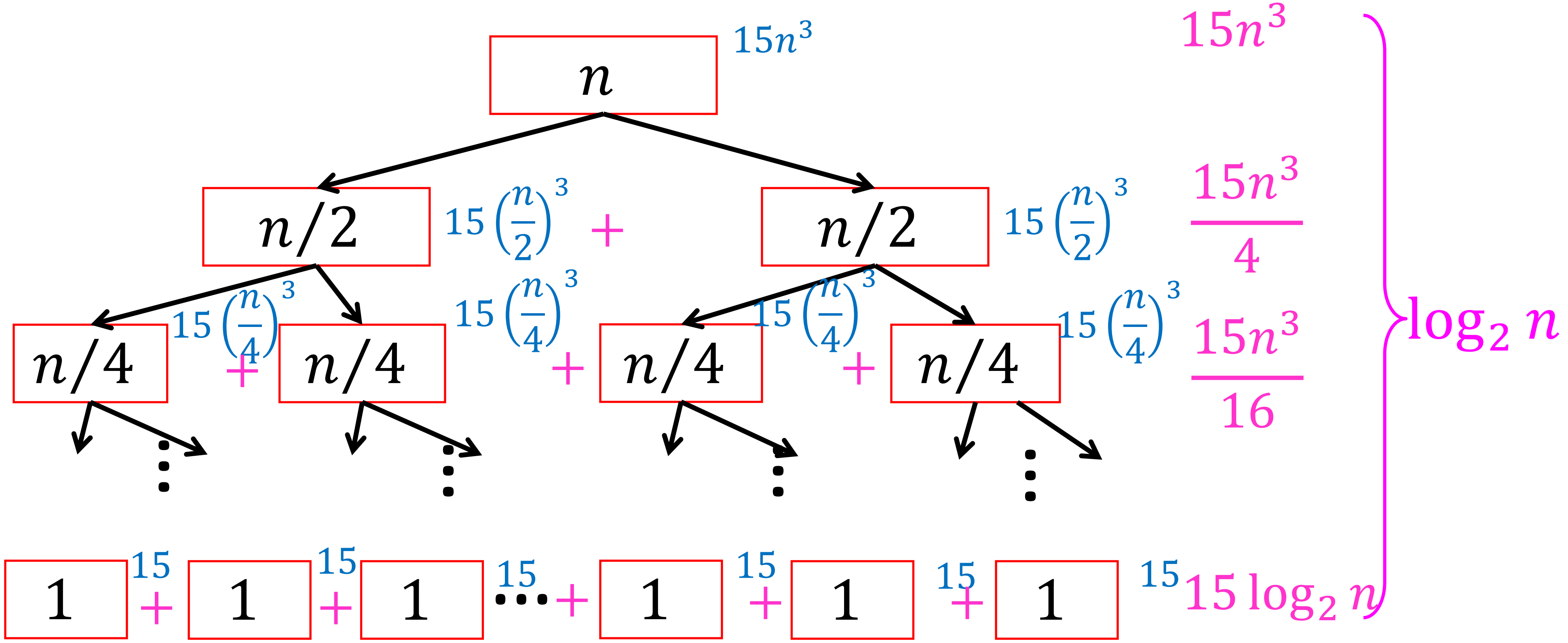
$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$

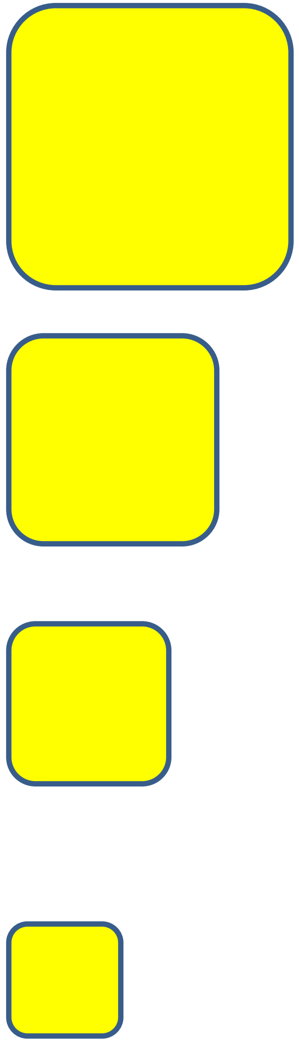
Case 3

$\Theta(n^3)$

Tree method

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





Recurrence Solving Techniques

Tree

Guess/Check

"Cookbook"

Substitution

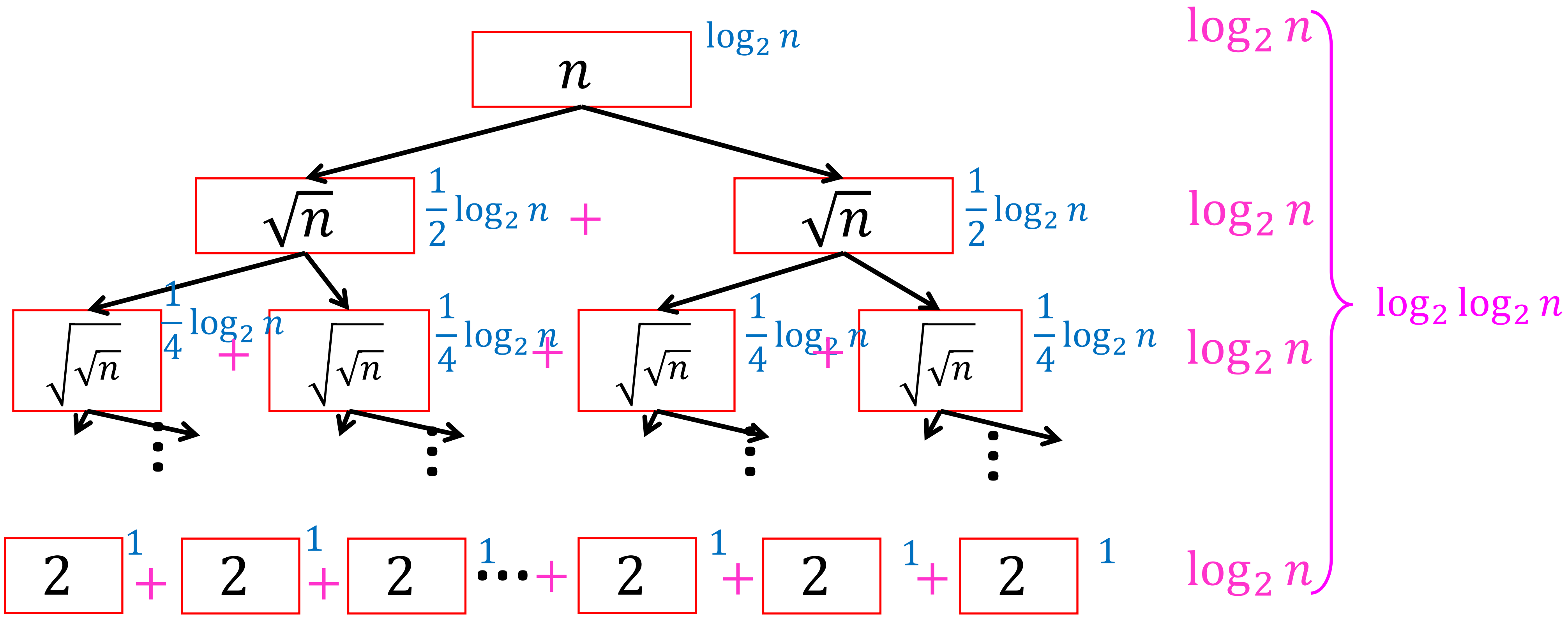
Substitution Method

- 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

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



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

Substitution Method

- Idea: take a "difficult" recurrence, re-express it such that one of our other methods applies.
- Example: $$T(n) = 2T(\sqrt{n}) + \log_2 n$$

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

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

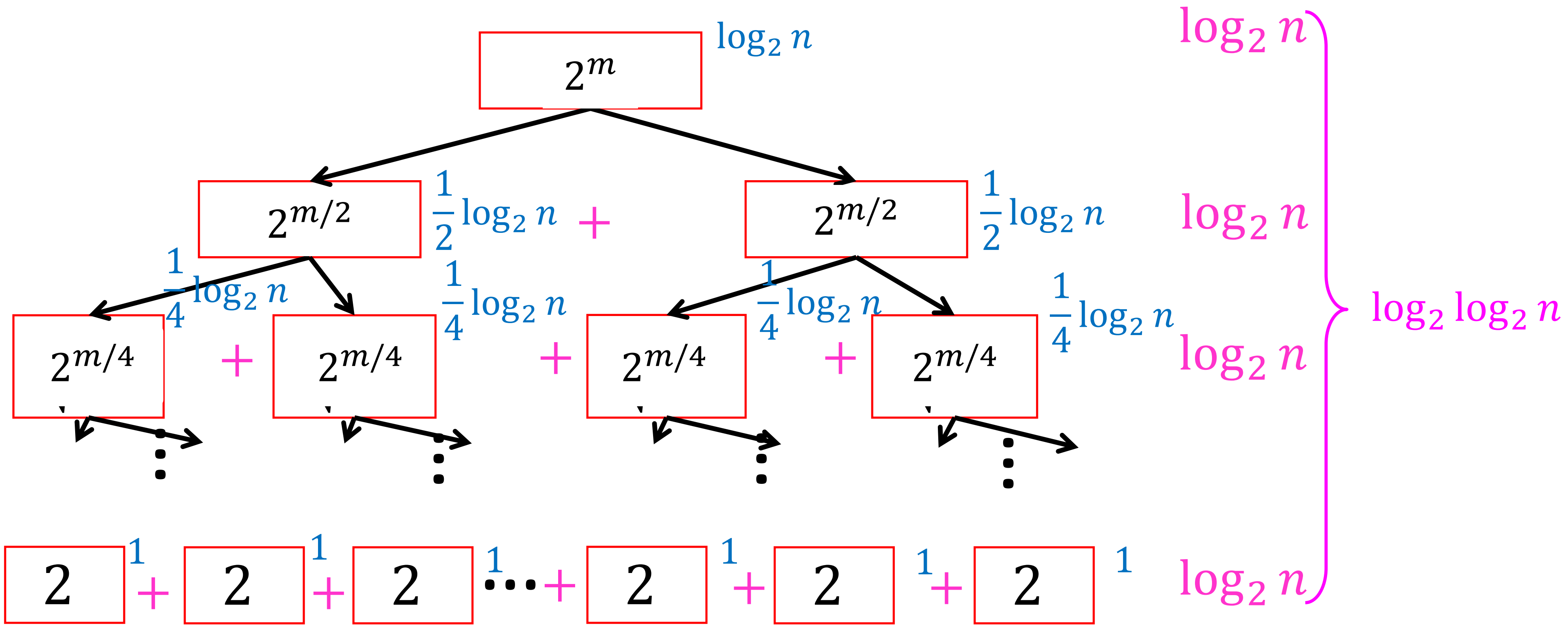
Let $S(m) = 2S\left(\frac{m}{2}\right) + m$ Case 2!

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

Let $\mathrm{T}(n) = \Theta(\log n \log \log n)$

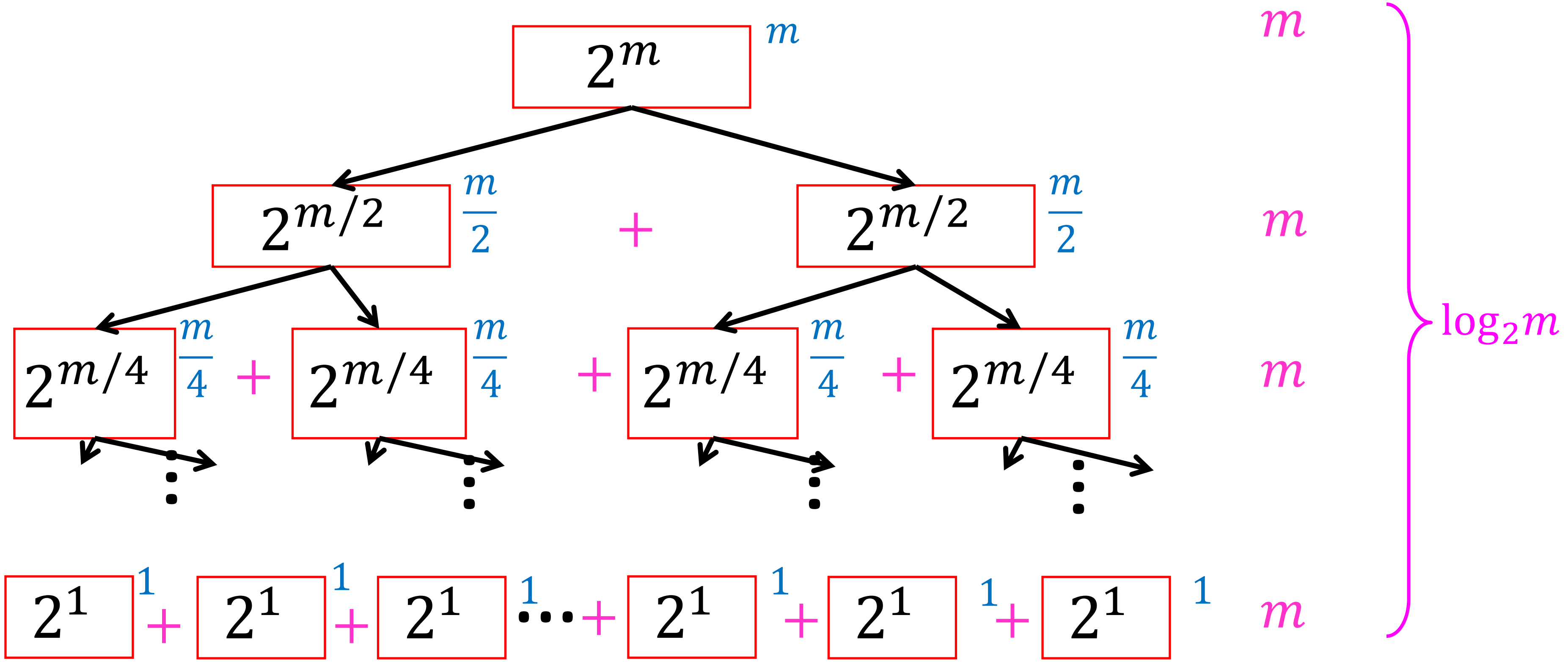
Tree method

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



Tree method

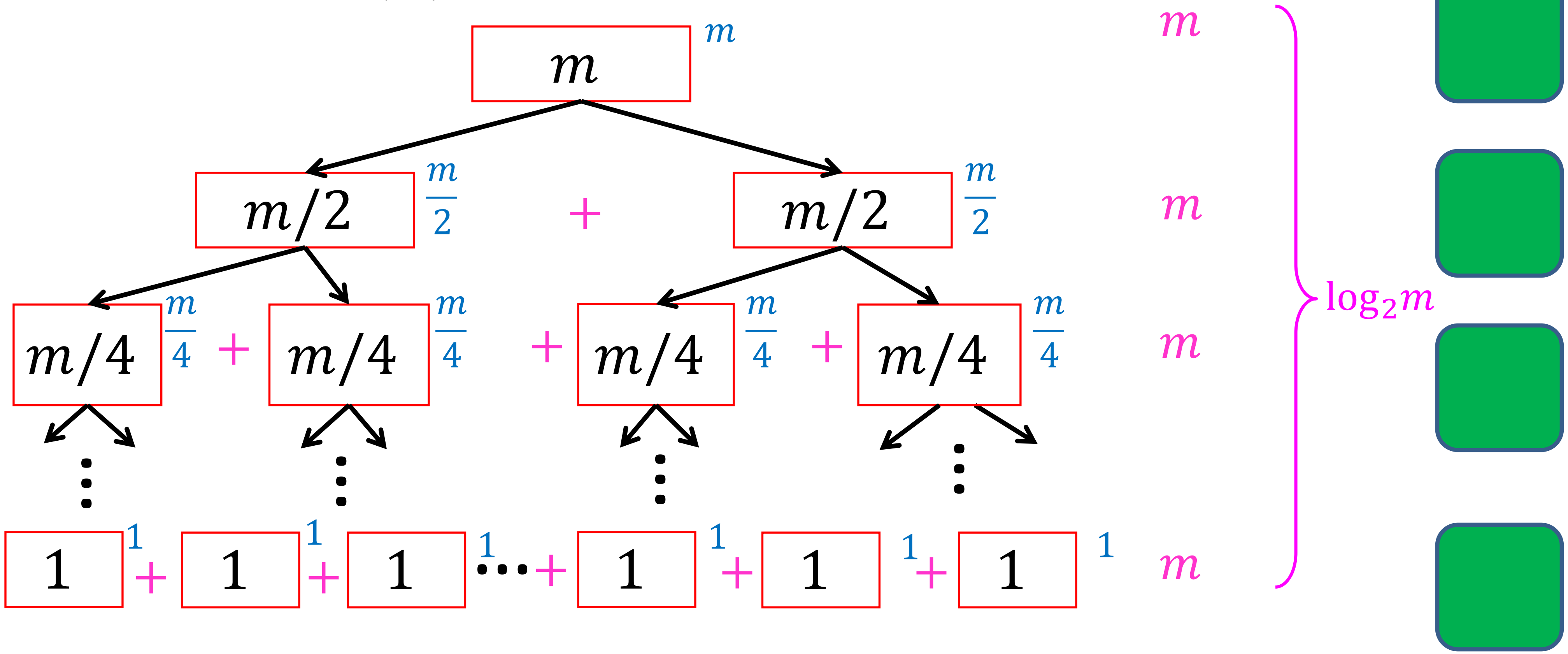
$n = 2^m$ $\qquad T(2^m) = 2T(2^{m/2}) + m$



Tree method

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

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



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