CS4102 Algorithms

Nate Brunelle

Spring 2018

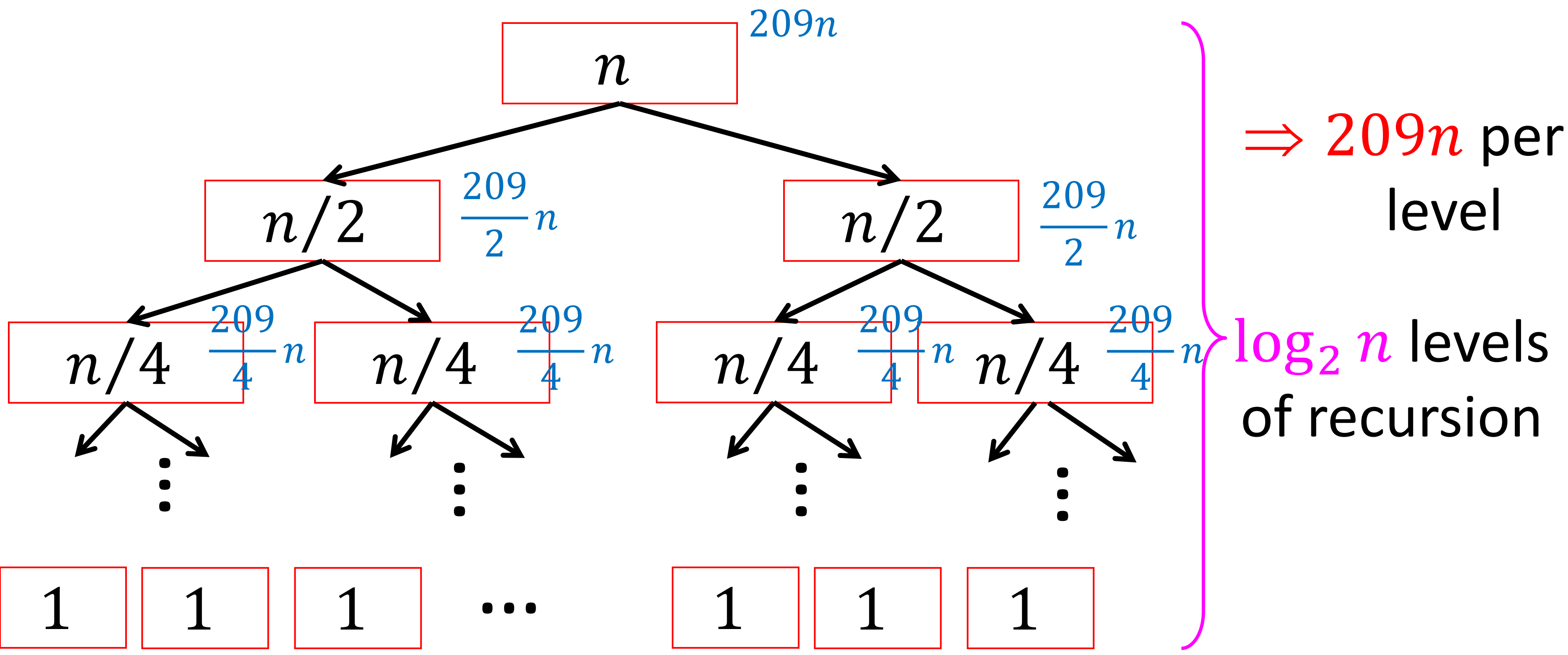
Warm Up

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

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

Tree method

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



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

Anonymous Feedback

Today’s Keywords

- Karatsuba
- Guess and check Method
- Induction

CLRS Readings

- Chapter 4

Homeworks

- Hw1 due 11pm Friday, February 2
  - Written (use Latex!)
  - Asymptotic notation
  - Recurrences
  - Divide and conquer

Merge Sort

- **Divide**:
  - Break n-element list into two lists of $^n/_2$ elements
- **Conquer**:
  - If $n > 1$:
    - Sort each sublist recursively
  - If $n = 1$:
    - List is already sorted (base case)
- **Combine**:
  - Merge together sorted sublists into one sorted list

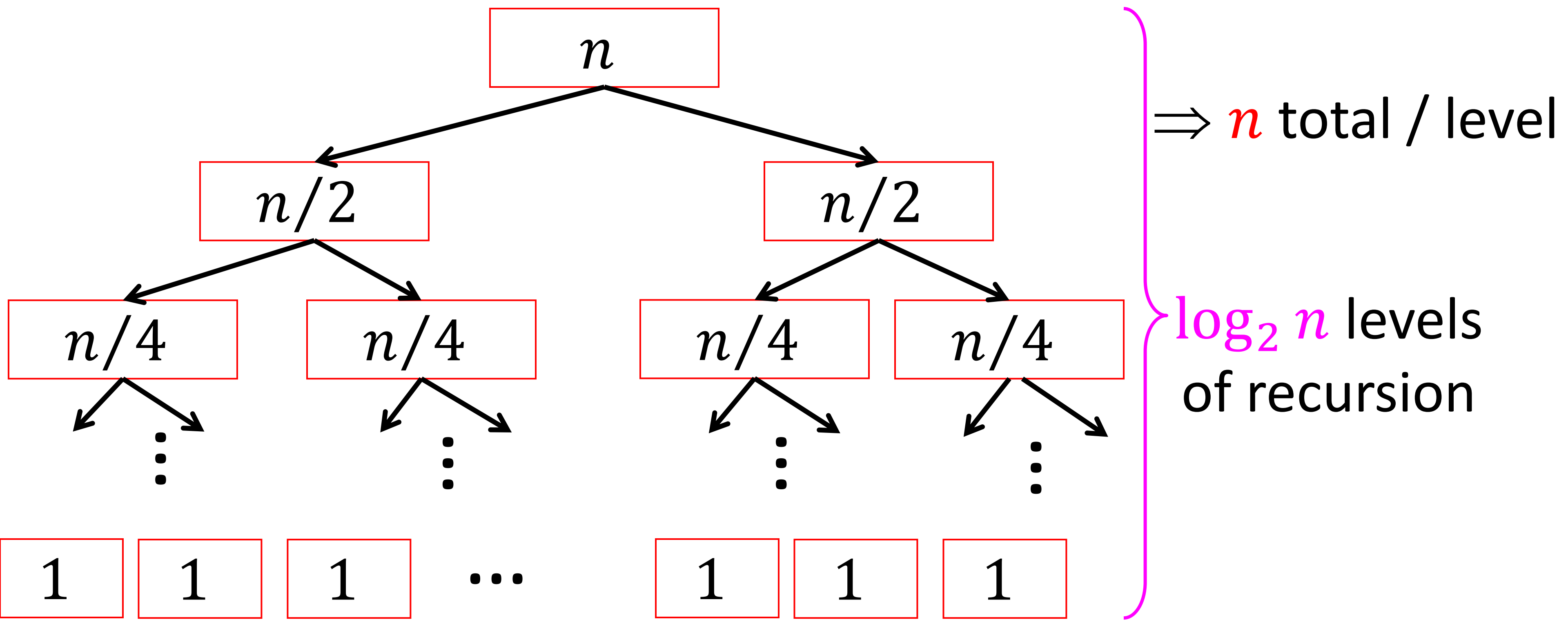
Analyzing Merge Sort

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

- **Divide**: 0 comparisons
- **Conquer**: recurse on 2 small problems, size $\frac{n}{2}$
- **Combine**: n comparisons
- **Recurrence:**
  - $T(n) = 2T(\frac{n}{2}) + n$

Tree method

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



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

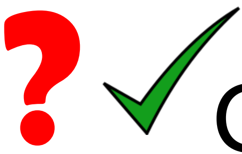
Analyzing Merge Sort

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

- **Divide**: 0 comparisons
- **Conquer**: recurse on 2 small problems, size $\frac{n}{2}$
- **Combine**: n comparisons
- **Recurrence:**
  - $T(n) = 2T(\frac{n}{2}) + n$

Recurrence Solving Techniques

Tree



Guess/Check

“Cookbook”

Substitution

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 $n = 1$:
    - Compute ac, ad, bc, bd directly (base case)
- **Combine**:
  - $10^{n}(ac) + 10^{\frac{n}{2}}(ad + bc) + bd$

Analyzing D&C Multiplication

1. Break into smaller subproblems

$$\begin{array}{r} \boxed{a}\,\boxed{b} \\ \times\ \boxed{c}\,\boxed{d} \\ \hline \end{array} \quad \begin{array}{l} = 10^{\frac{n}{2}}\,\boxed{a} + \boxed{b} \\ = 10^{\frac{n}{2}}\,\boxed{c} + \boxed{d} \end{array}$$

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

Divide and Conquer method

2. Use recurrence relation to express recursive running time

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

Recursively solve

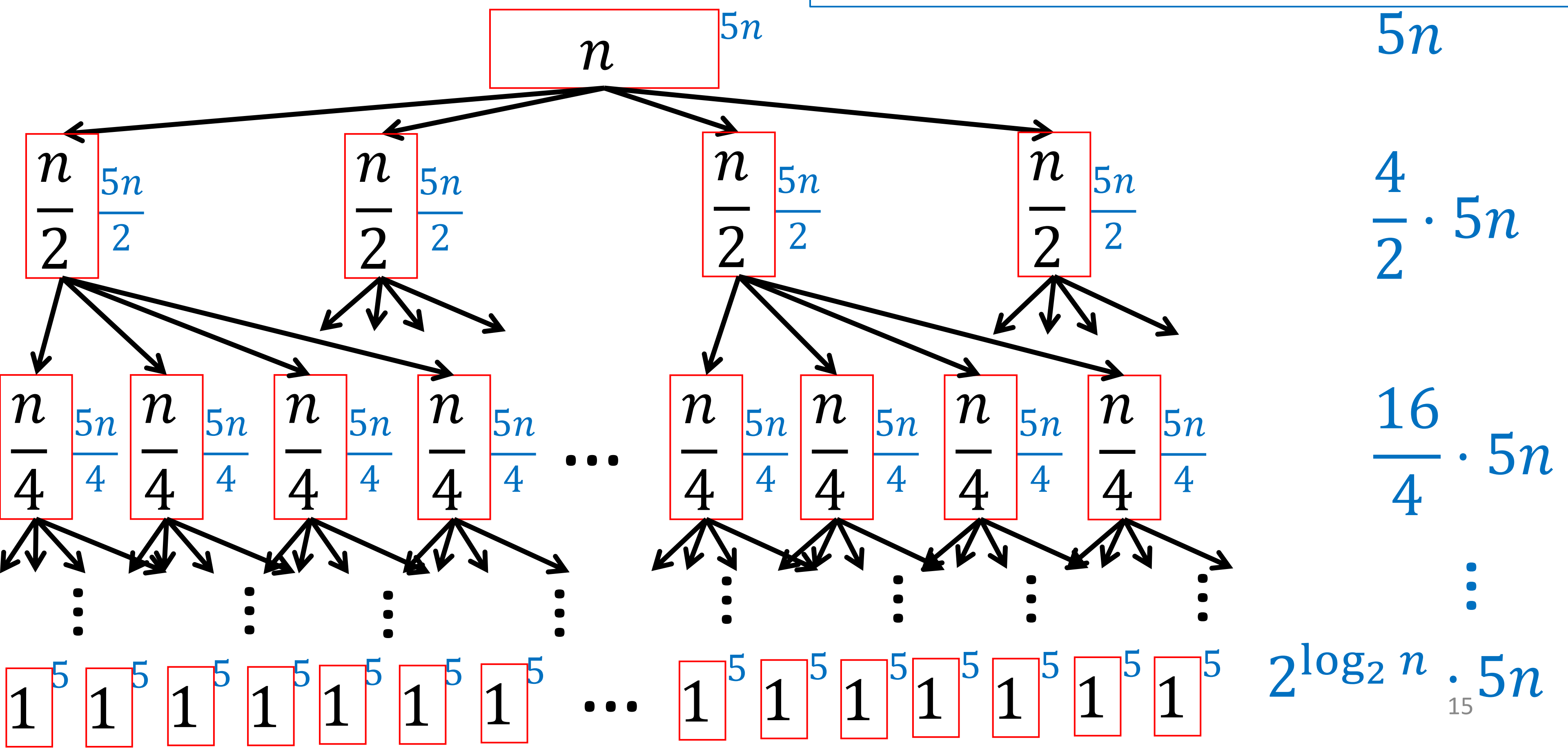
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Divide and Conquer method

3. Use asymptotic notation to simplify

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

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



Karatsuba

1. Break into smaller subproblems

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

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

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

Karatsuba

$$100^2(\boxed{ac}) + 100(\boxed{ad + bc}) + \boxed{bd}$$

Can't avoid these

This can be simplified

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

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

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

Two multiplications

One multiplication

Karatsuba

2. Use recurrence relation to express recursive running time

$$100^2\boxed{(ac)} + 100\left(\boxed{(a+b)(c+d)} - ac - bd\right) + \boxed{bd}$$

Recursively solve

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

Karatsuba

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

a b

× c d

Karatsuba Algorithm

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

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

Pseudo-code

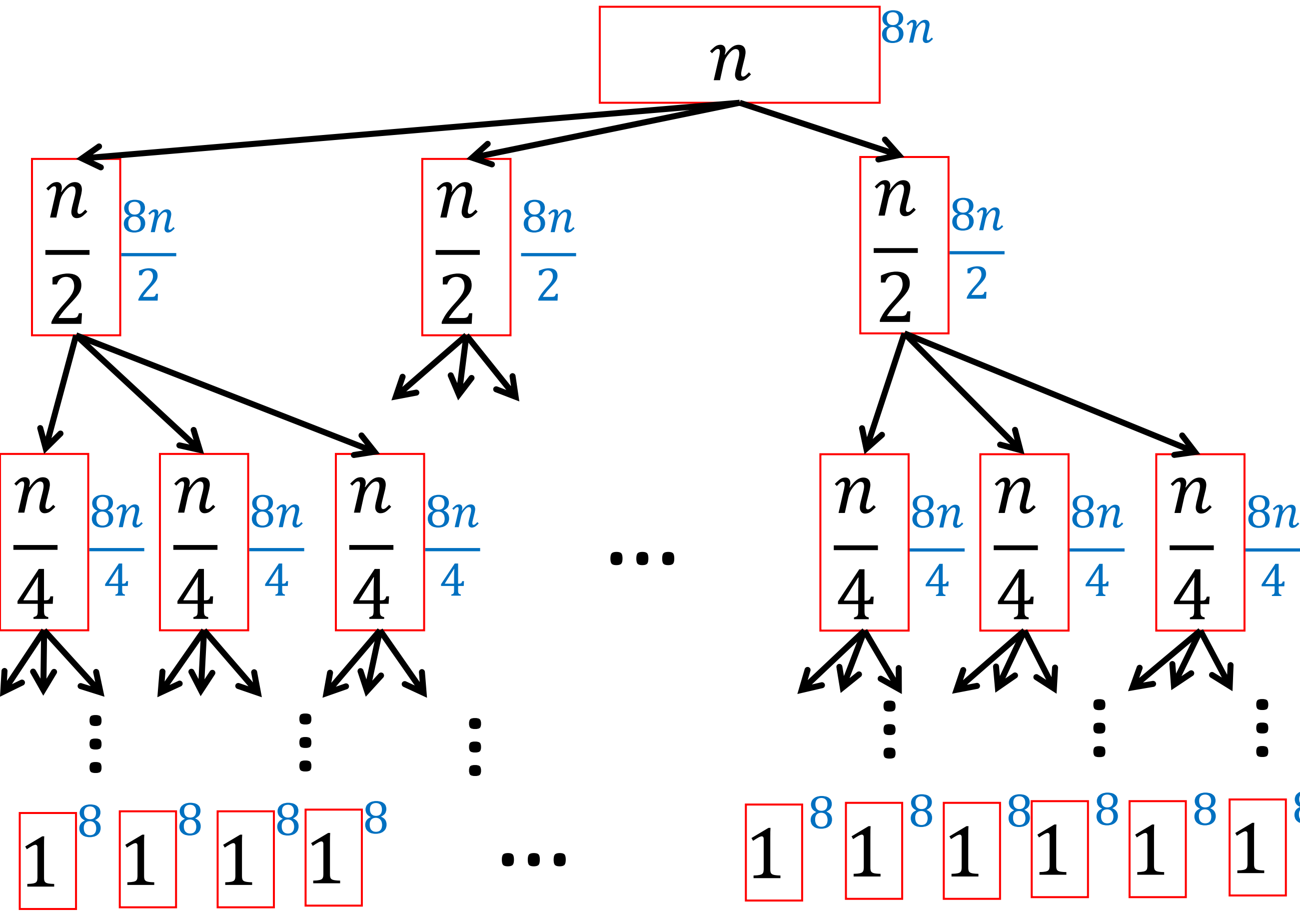
1. x = Karatsuba(a,c)
2. y = Karatsuba(a,d)
3. z = Karatsuba(a+b,c+d)-x-y
4. Return 10^nx +$10^{n/2}$z+y

Karatsuba

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

3. Use asymptotic notation to simplify

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



$$8 \cdot 1n$$

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

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

$$\vdots$$

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

Karatsuba

3. Use asymptotic notation to simplify

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

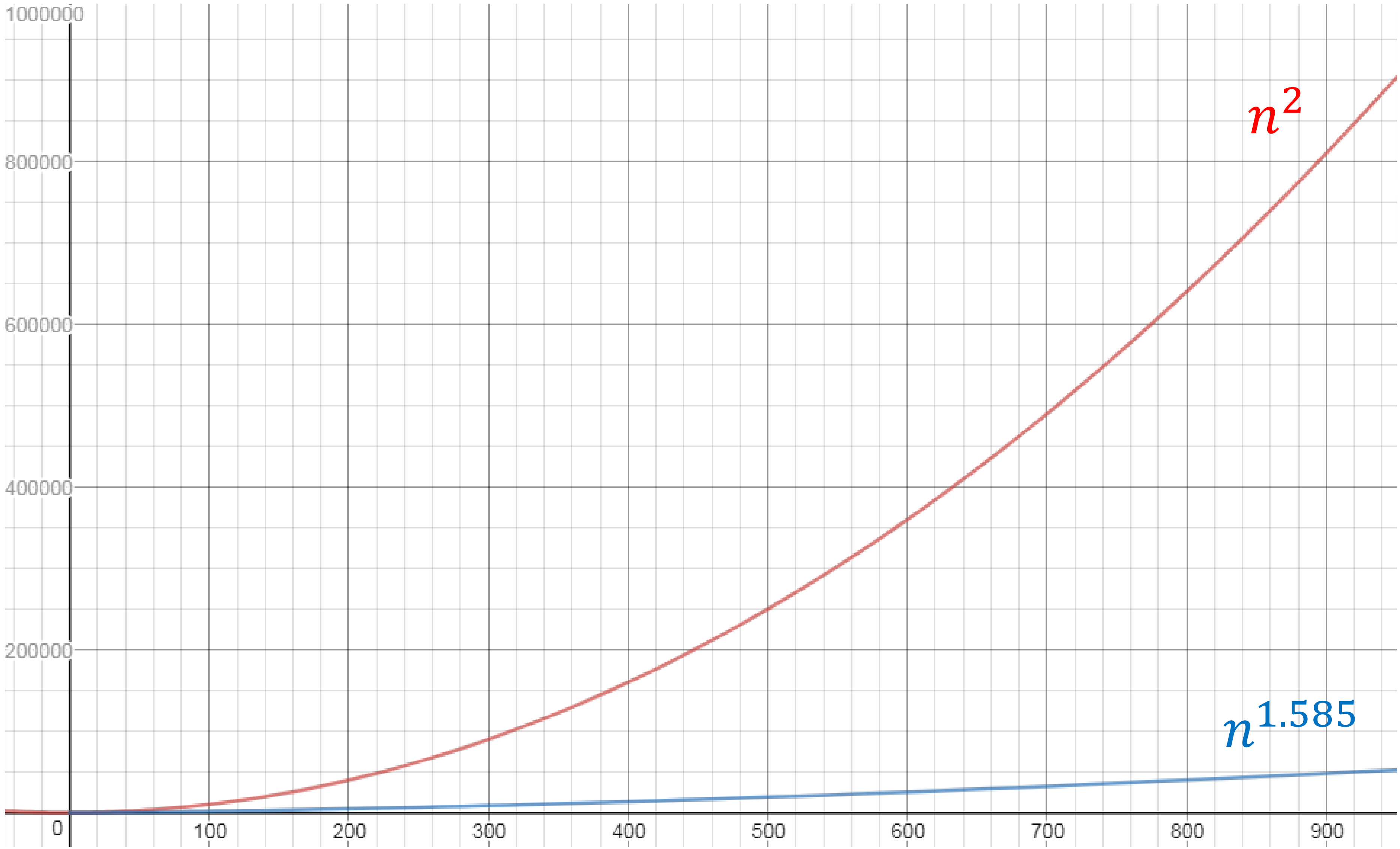
$$T(n) = 8n \sum_{i=0}^{\log_2 n} \left({}^{3}/_{2}\right)^{i}$$

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

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

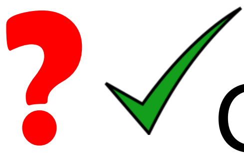
$$T(n) = 24\left(n^{\log_2 3}\right) - 16n \;= \Theta\left(n^{\log_2 3}\right)$$

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


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 cases: $P(1)$ holds

Hypothesis: $\forall n < n_0, P(n)$ holds

Inductive step: $P(n_0) \Rightarrow P(n_0 + 1)$

Guess and Check Intuition

- To Prove: $T(n) = O(g(n))$
- Consider: $g_*(n) = O(g(n))$
- Goal: show $\exists n_0$ s.t. $\forall n > n_0, T(n) < g_*(n)$
- Technique: Induction
  - Base cases:
    - show $T(1) < g_*(1), T(2) < g_*(2)$, ... for a small number of cases
  - Hypothesis:
    - $\forall n < n_0, T(n) < g_*(n)$
  - Inductive step:
    - $T(n_0 + 1) < g_*(n_0 + 1)$

Karatsuba Guess and Check (Loose)

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

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

Base cases: $T(1) = 8 < 3000$

$T(2) = 3(8) + 16 = 40 < 3000 \cdot 2^{1.6}$

… up to some small k

Hypothesis: $\forall n < n_0\; T(n) < 3000n^{1.6}$

Inductive step: $T(n_0 + 1) < 3000(n_0 + 1)^{1.6}$

see lecture supplemental

Mergesort Guess and Check

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

Goal: $T(n) < 2n \log n \; = O(n \log n)$

Base cases: $T(1) = 0$

$T(2) = 2 < 4 \log 2$

… up to some small k

Hypothesis: $\forall n < n_0 \, T(n) < n \log n$

Inductive step: $T(n_0 + 1) < 2(n_0 + 1) \log(n_0 + 1)$

[see lecture supplemental]

Karatsuba Guess and Check

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

Goal: $T(n) < n^{\log_2 3} - 16n = O(n^{\log_2 3})$

Base cases: by inspection, holds for small n (at home)

Hypothesis: $\forall n < n_0 \; T(n) < n^{\log_2 3} - 16n$

Inductive step: $T(n_0 + 1) < (n_0 + 1)^{\log_2 3} - 16(n_0 + 1)$

see lecture supplemental

What if we leave out the $-16n$?

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

Goal: $T(n) < n^{\log_2 3} - 16n = O(n^{\log_2 3})$

Base cases: by inspection, holds for small n (at home)

Hypothesis: $\forall n < n_0 \; T(n) < n^{\log_2 3} - 16n$

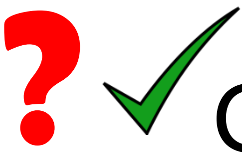
Inductive step: $T(n_0 + 1) < (n_0 + 1)^{\log_2 3} - 16(n_0 + 1)$

What we wanted: $T(n_0 + 1) < (n_0 + 1)^{\log_2 3}$ Induction failed!

What we got: $T(n_0 + 1) < (n_0 + 1)^{\log_2 3} + 8(n_0 + 1)$

Recurrence Solving Techniques

Tree



Guess/Check

“Cookbook”

Substitution

Observation

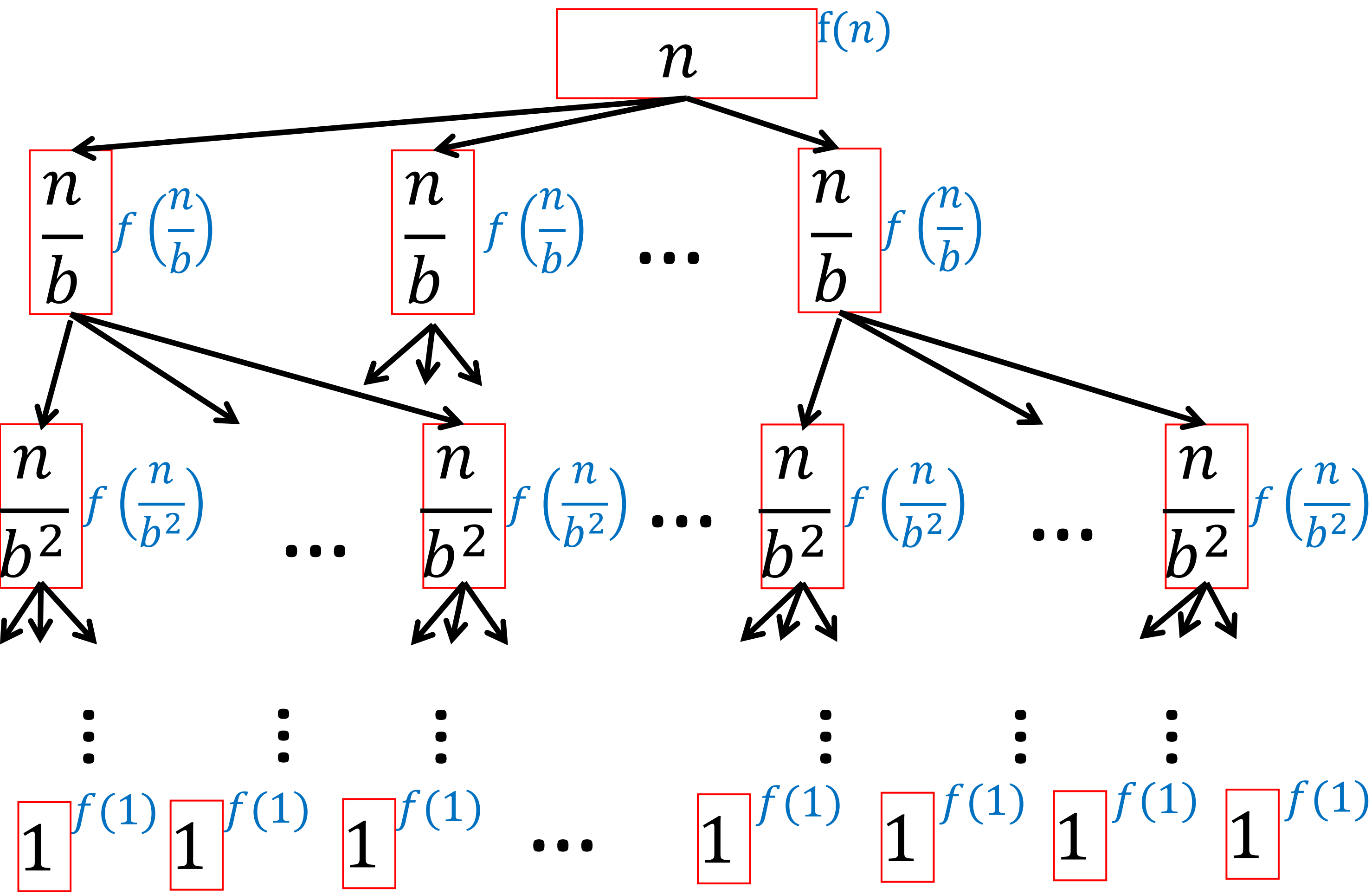
- **Divide**: $D(n)$ time,
- **Conquer**: recurse on small problems, size s
- **Combine**: $\mathrm{C}(n)$ time
- **Recurrence:**
  - $T(n) = D(n) + \sum T(s) + C(n)$

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

General

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

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



$$f(n)$$

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

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

$$\vdots$$

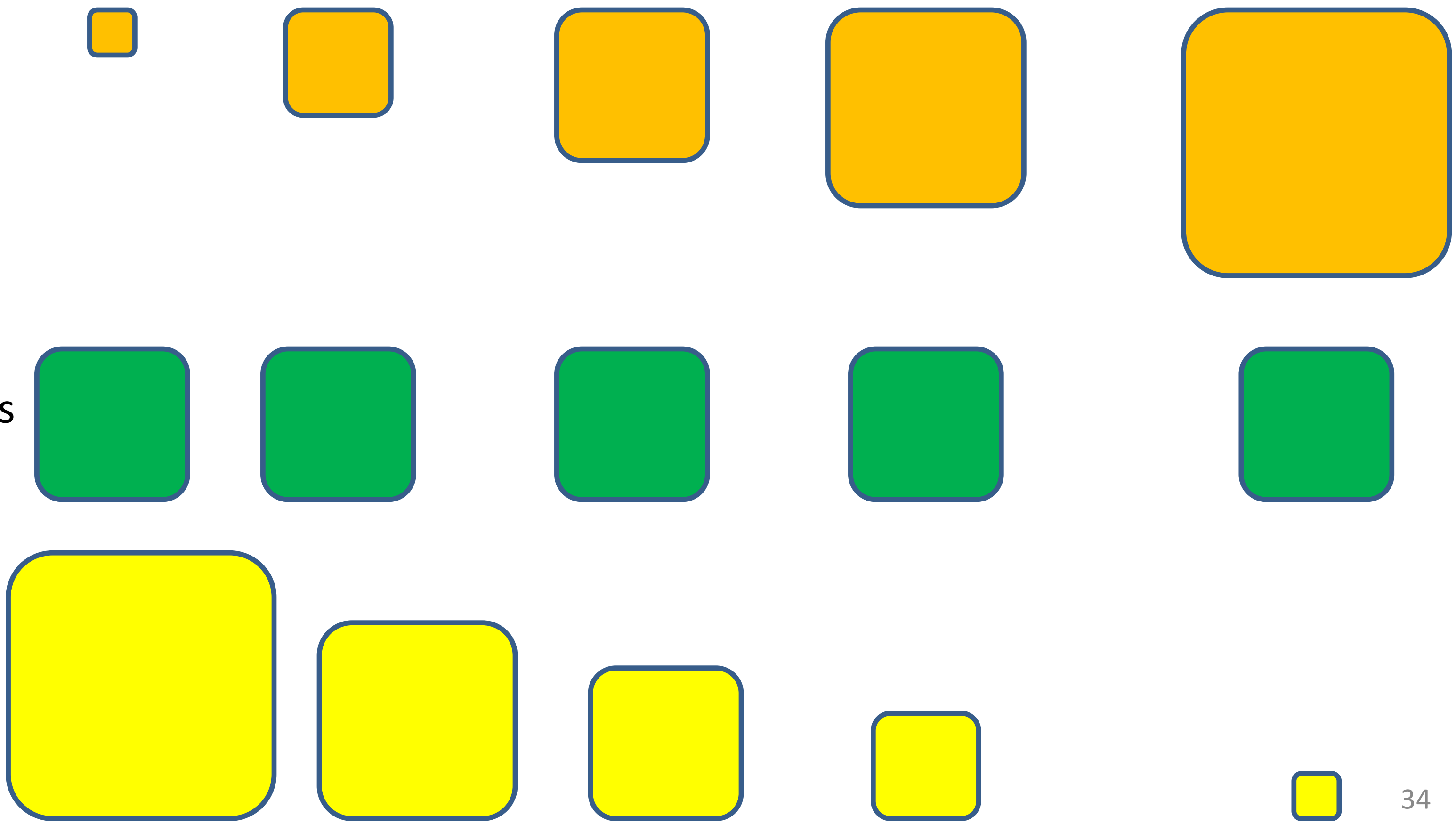
$$a^{\log_b n} f(1)$$

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

Case 1:
Most works
happens at
the leaves

Case 2:
Work happens
consistently
throughout

Case 3:
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

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

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

Insert math here…

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

see lecture supplemental

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

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

$$T(n) = 4T\left(\frac{n}{2}\right) + 8n$$

Case 1

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

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