CS4102 Algorithms

Nate Brunelle

Fall 2017

Warm up

Simplify:

$$(1 + a + a^2 + a^3 + a^4 + \cdots + a^L)(a - 1) = ?$$

$$(a + a^2 + a^3 + a^4 + a^5 + \cdots + a^L + a^{L+1}) +$$
$$(-a - a^2 - a^3 - a^4 - a^5 - \cdots - a^L - 1) =$$
$$a^{L+1} - 1$$

$$\sum_{i=0}^{L} a^i = \frac{a^{L+1} - 1}{a - 1}$$

Anonymous Feedback

- Suggestion: within the lecture slides folder, make two separate folders: one for PDFs and one for PowerPoints, so we don't have to look at 2 of every slidedeck. So far so good. Keep up the good work Nate! :)
- I thought this attendence thing was gonna be a s***fest, but it was actually pretty efficient –KL
- What do you mean caricature? Because I can't draw

Today’s Keywords

- Divide and Conquer
- Recurrences
- Merge Sort
- Karatsuba
- Tree Method
- Guess and check Method
- Induction

CLRS Readings

- Chapter 4

Homeworks

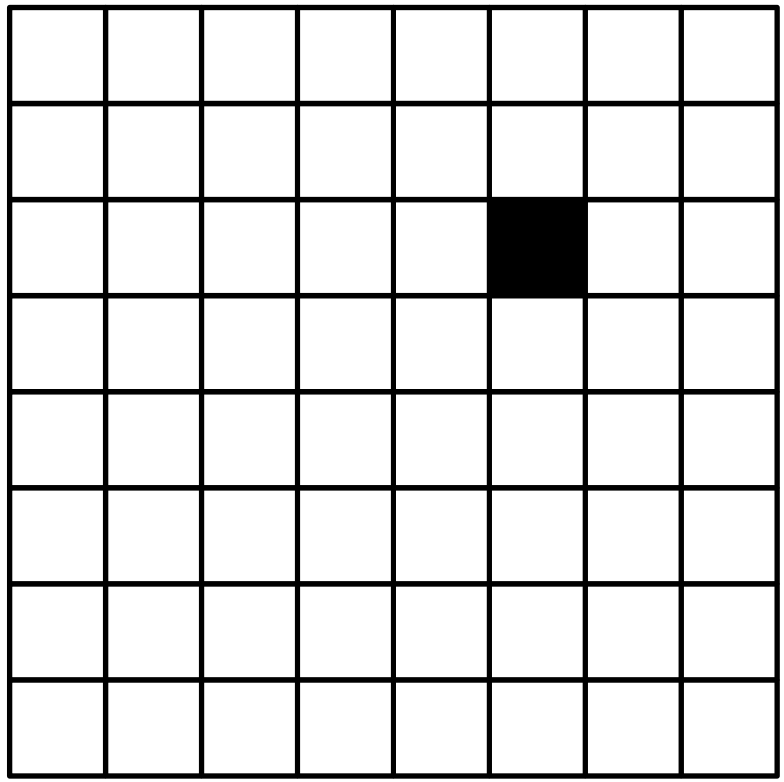
- Hw0 due 11pm Friday, January 26
  - Submit BOTH pdf and zip!
  - Use align environment

$$
\begin{aligned}
a &= 2c \\
a^2 &= (2c)^2 \\
2b^2 &= 4c^2
\end{aligned}
$$

- Hw1 released Friday, January 26
  - Due 11pm Friday, February 2
  - Written (use Latex!)
  - Asymptotic notation
  - Recurrences
  - Divide and conquer

Trominoes Puzzle Solution

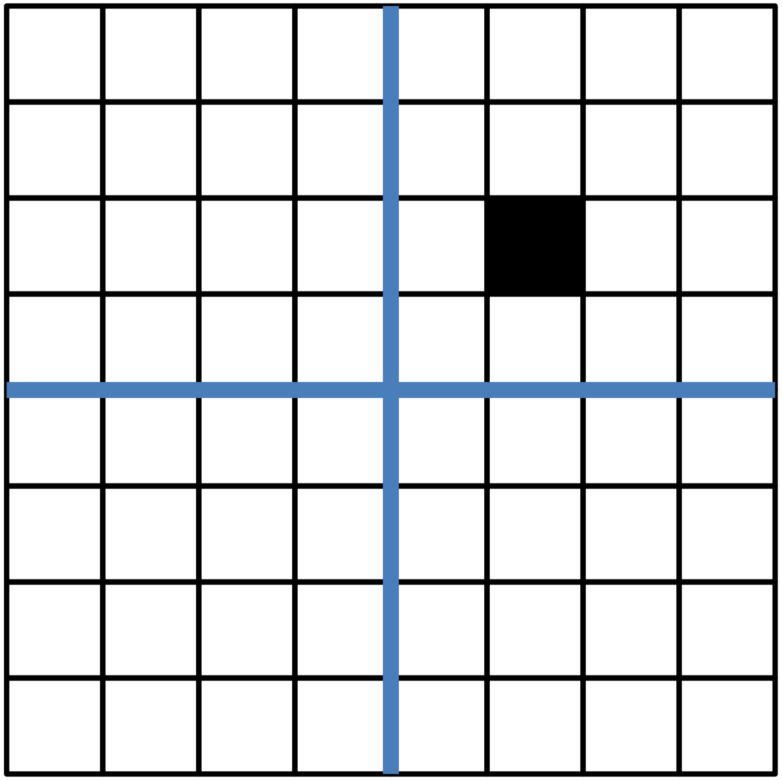
2^n



2^n

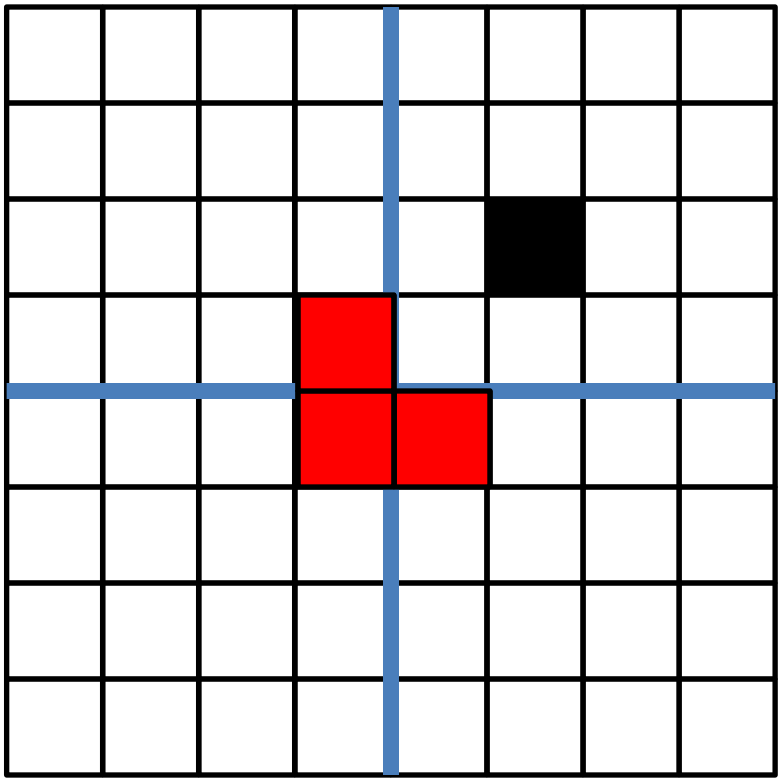
What about larger boards?

Trominoes Puzzle Solution



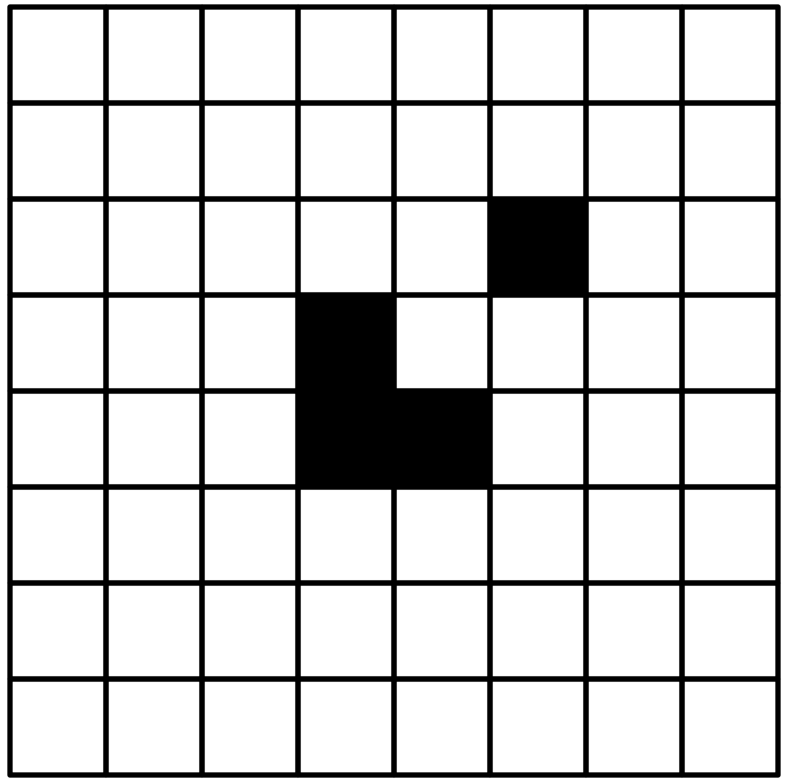
Divide the board into quadrants

Trominoes Puzzle Solution



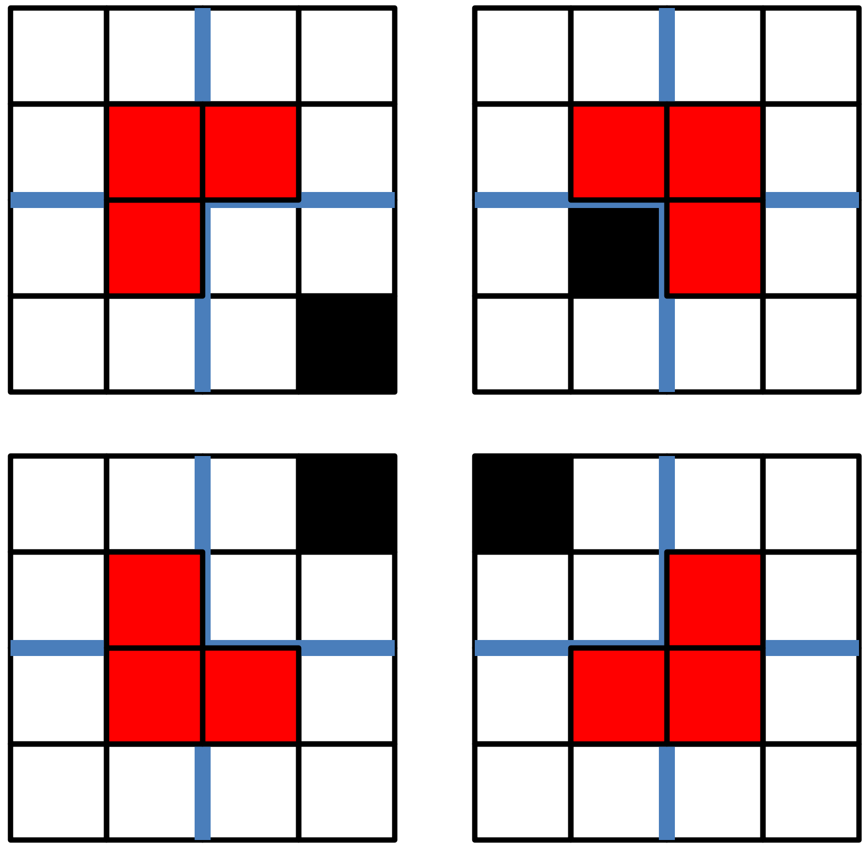
Place a tromino to occupy the three quadrants without the missing piece

Trominoes Puzzle Solution



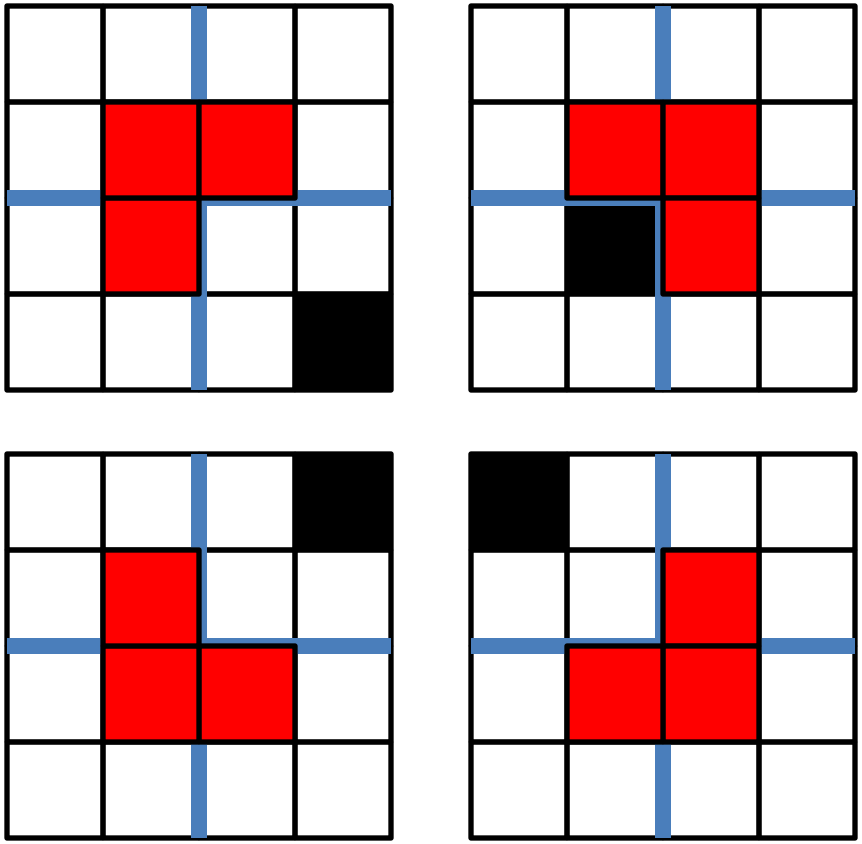
Each quadrant is now a smaller subproblem

Trominoes Puzzle Solution



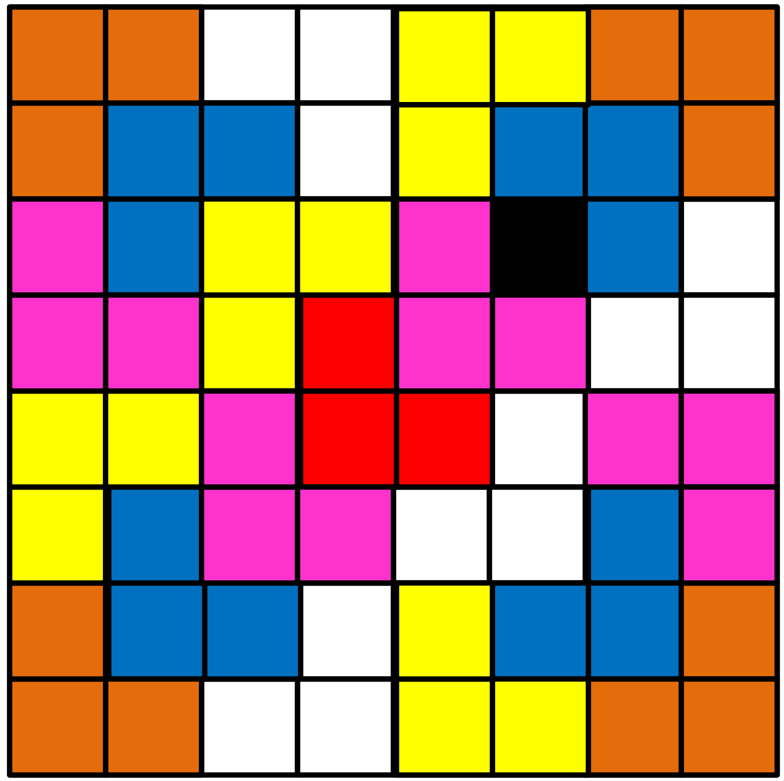
Solve **Recursively**

Divide and Conquer



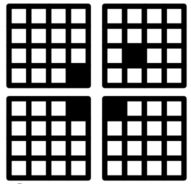
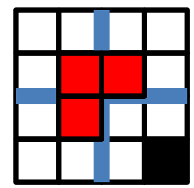
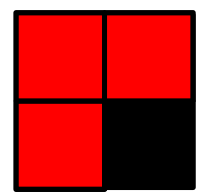
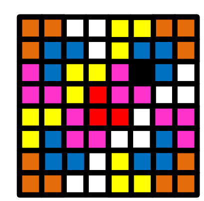
Our first algorithmic technique!

Trominoes Puzzle Solution



Divide and Conquer*

When is this a good strategy?

- **Divide**: 
  - Break the problem into multiple subproblems, each smaller instances of the original
- **Conquer**:
  - If the subproblems are "large": 
    - Solve each subproblem recursively
  - If the subproblems are "small": 
    - Solve them directly (base case)
- **Combine**:
  - Merge together solutions to subproblems 

*CLRS Chapter 4

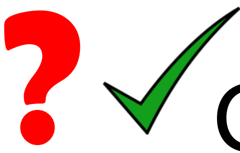
Analyzing Divide and Conquer

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

Recurrence Solving Techniques

Tree



Guess/Check

"Cookbook"

Substitution

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

Merge

- **Combine**: Merge sorted sublists into one sorted list
- We have:
  - 2 sorted lists (L_1, L_2)
  - 1 output list (L_{out})

While (L_1 and L_2 not empty):

  If $L_1[0] \leq L_2[0]$:

    L_{out}.append(L_1.pop())

  Else:

    L_{out}.append(L_2.pop())

L_{out}.append(L_1)

L_{out}.append(L_2)

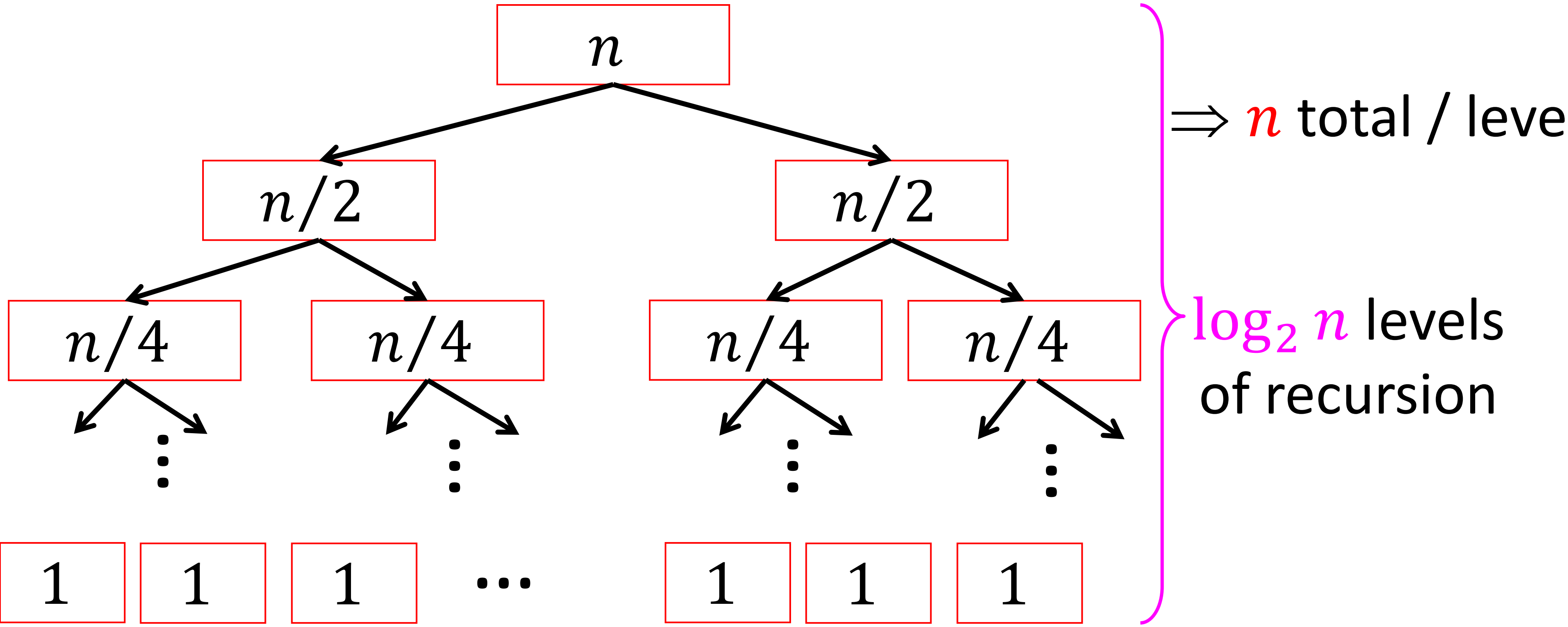
Analyzing Merge Sort

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

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



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

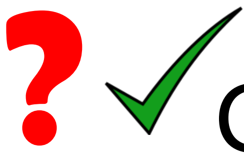
Analyzing Merge Sort

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



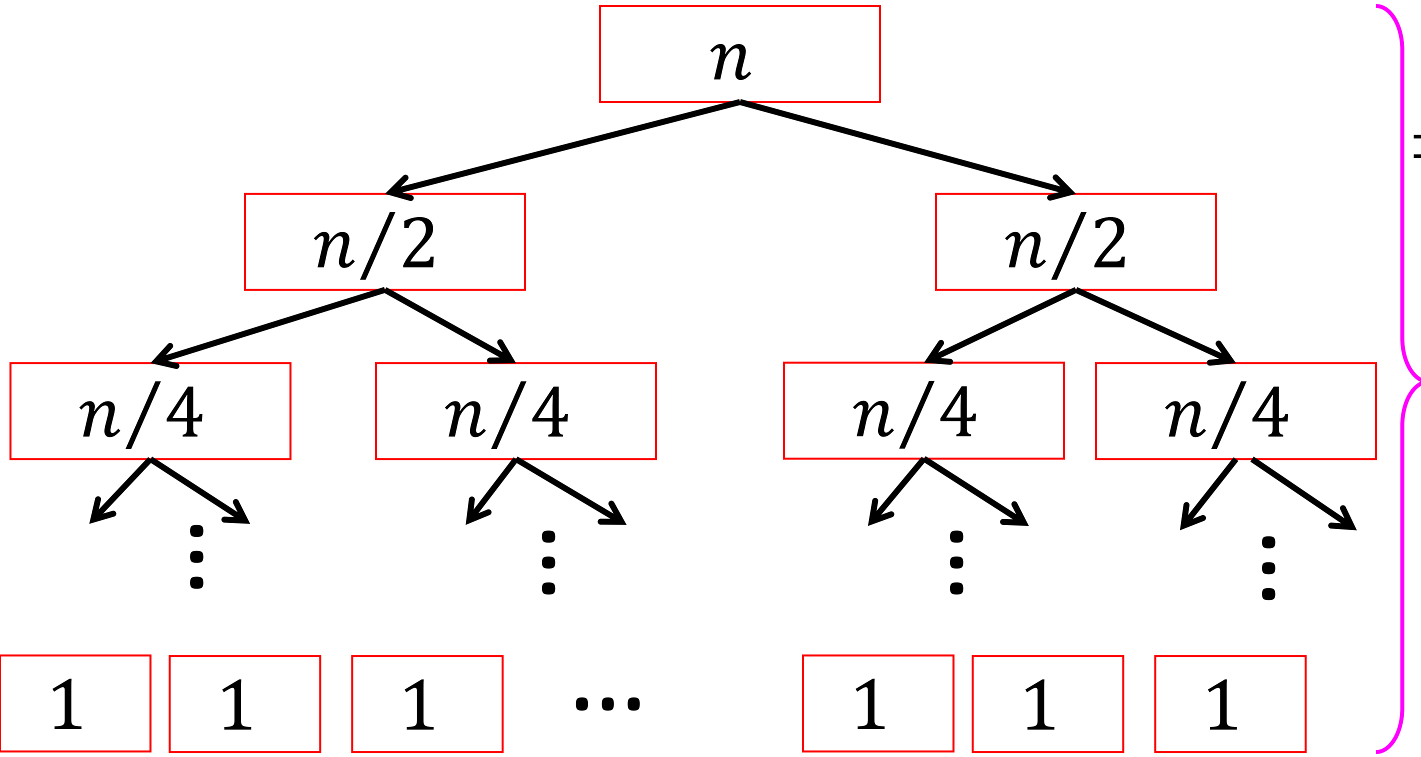
Guess/Check

"Cookbook"

Substitution

Tree method

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



$\Rightarrow$ n total / level

$\log n$ levels of recursion

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

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

- What makes a "good" algorithm?
- How do we measure input size?
- What do we "count" for run time?

"Schoolbook" Method

How many total multiplications?

n-digit numbers

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

n mults

n mults

n mults

n mults

n levels

$\Rightarrow \theta(n^2)$

Divide and Conquer method

1. Break into smaller subproblems

$$\boxed{a}\,\boxed{b} = 100\,\boxed{a} + \boxed{b}$$

$$\times\ \boxed{c}\,\boxed{d} = 100\,\boxed{c} + \boxed{d}$$

$$100^2\left(\boxed{a}\times\boxed{c}\right) +$$

$$100\ \left(\boxed{a}\times\boxed{d} + \boxed{b}\times\boxed{c}\right) +$$

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

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 $n = 1$:
    - Compute ac, ad, bc, bd directly (base case)
- **Combine**:
  - $100^2(ac) + 100(ad + bc) + bd$

Divide and Conquer method

2. Use recurrence relation to express recursive running time

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

Recursively solve

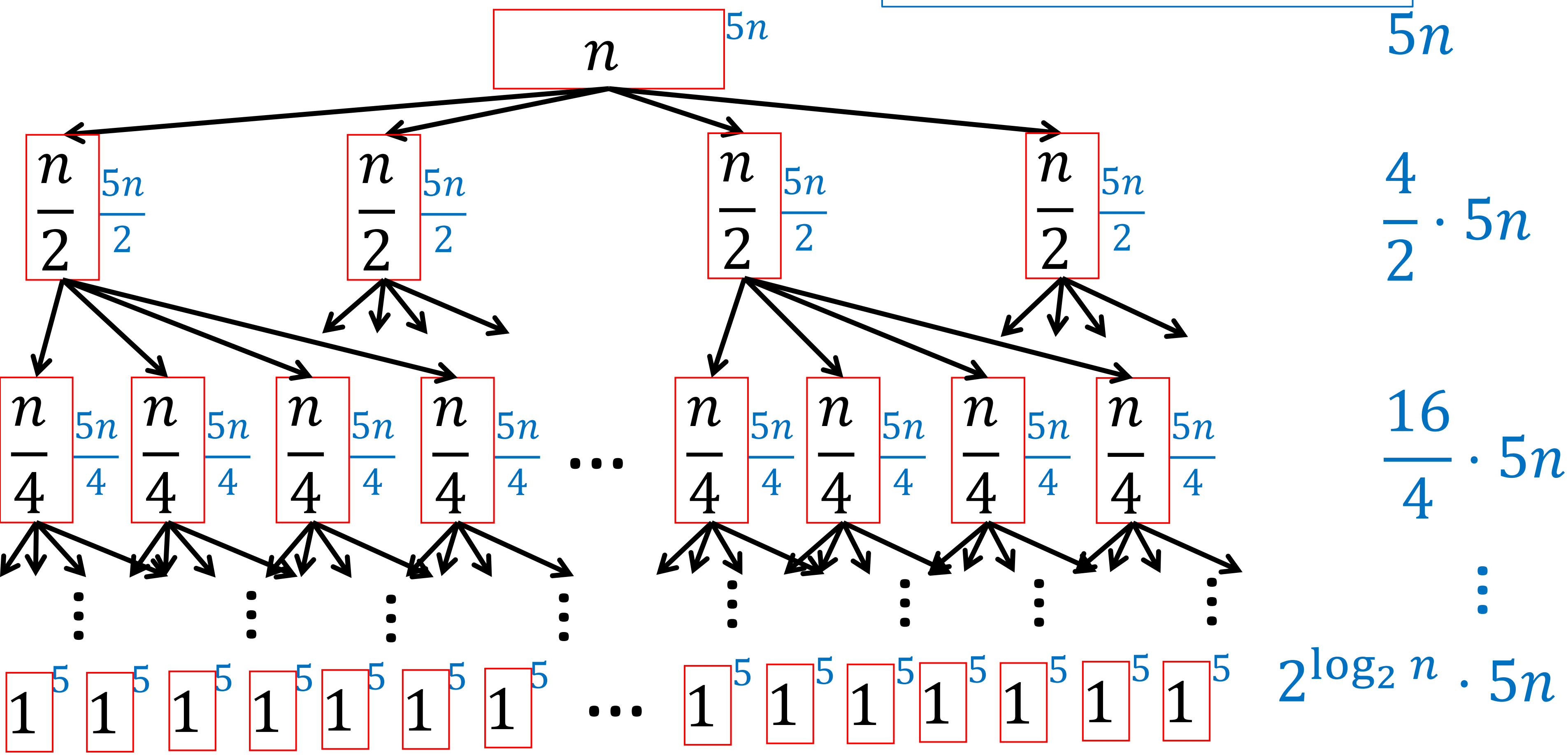
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Divide and Conquer method

3. Use asymptotic notation to simplify

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

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



Divide and Conquer method

3. Use asymptotic notation to simplify

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

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

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

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

Karatsuba

1. Break into smaller subproblems

$$\begin{array}{r} \boxed{a}\boxed{b} \\ \times\ \boxed{c}\boxed{d} \\ \hline \end{array} \quad \begin{array}{l} = 100\,\boxed{a} + \boxed{b} \\ = 100\,\boxed{c} + \boxed{d} \end{array}$$

$$100^2\left(\boxed{a} \times \boxed{c}\right) +$$
$$100\,\left(\boxed{a} \times \boxed{d} + \boxed{b} \times \boxed{c}\right) +$$
$$\left(\boxed{b} \times \boxed{d}\right)$$

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

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

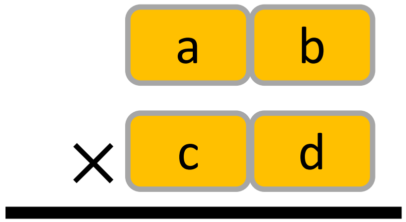
$$100^2\boxed{(ac)} + 100\left(\boxed{(a+b)(c+d)} - ac - bd\right) + \boxed{bd}$$

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



Karatsuba Algorithm

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

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

<u>Pseudo-code</u>

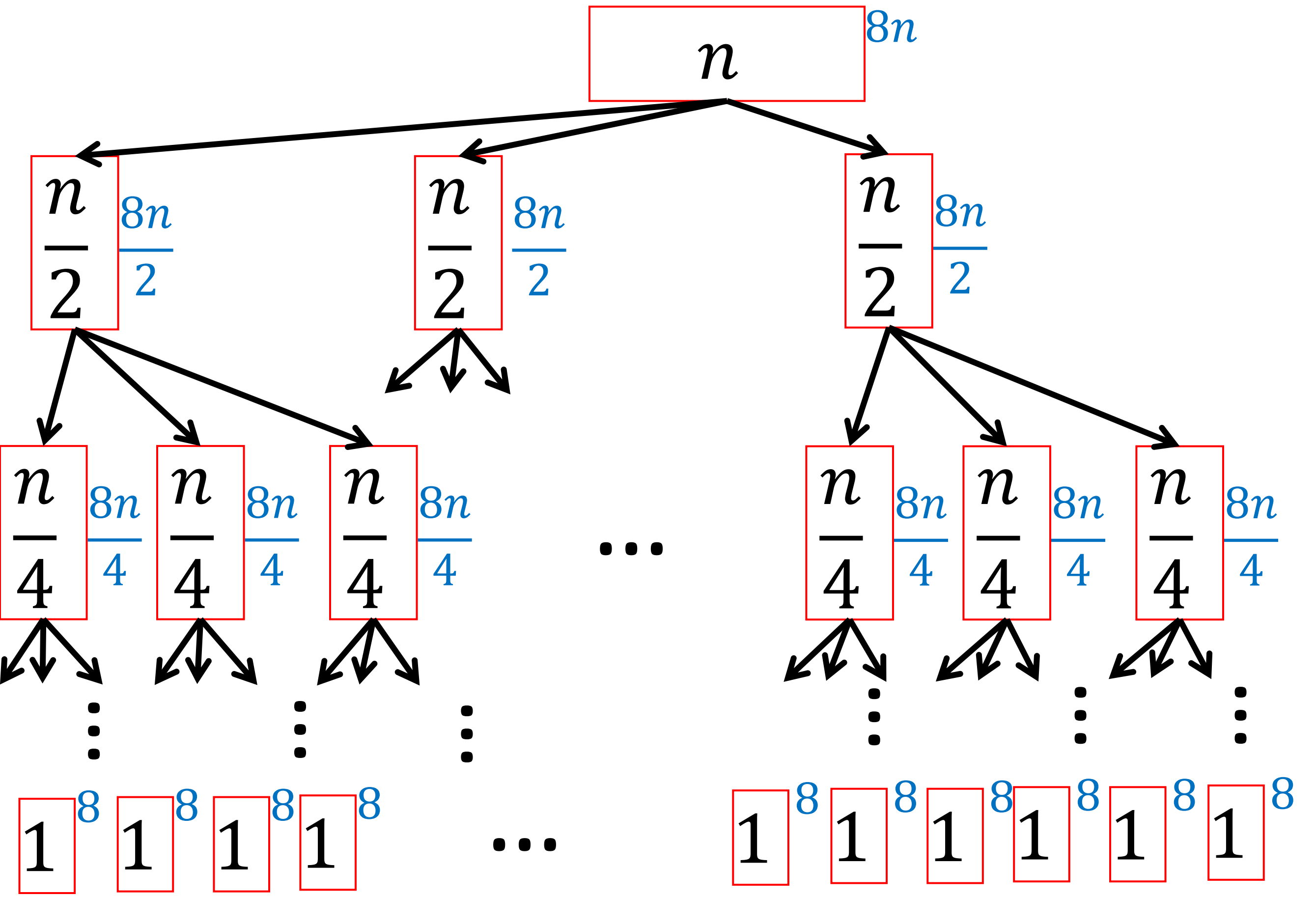
1. x = Karatsuba(a,c)
2. y = Karatsuba(a,d)
3. z = Karatsuba(a+b,c+d)-x-y
4. Return 10^nx +$10^{n/2}$z+y

Karatsuba

3. Use asymptotic notation to simplify

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

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



$$8 \cdot 1n$$

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

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

$$\vdots$$

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

Karatsuba

3. Use asymptotic notation to simplify

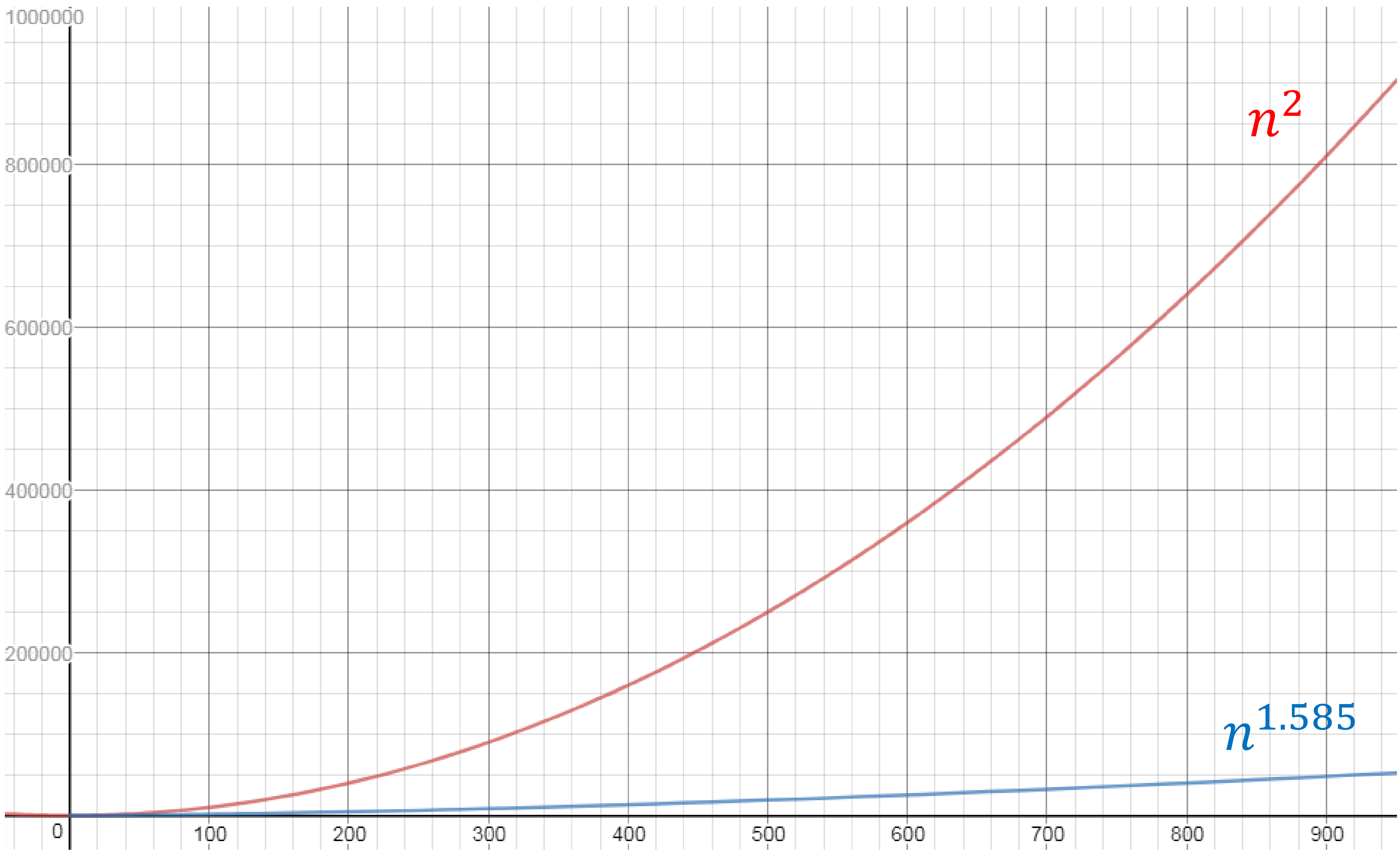
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

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

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

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

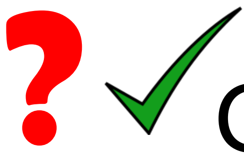
$$T(n) = 24\left(n^{\log_2 3}\right) - 16n \;\; = \Theta\left(n^{\log_2 3}\right)$$
$$\approx \Theta\left(n^{1.585}\right)$$



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

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