CS 3100
Data Structures and Algorithms 2
Lecture 7: Divide and Conquer

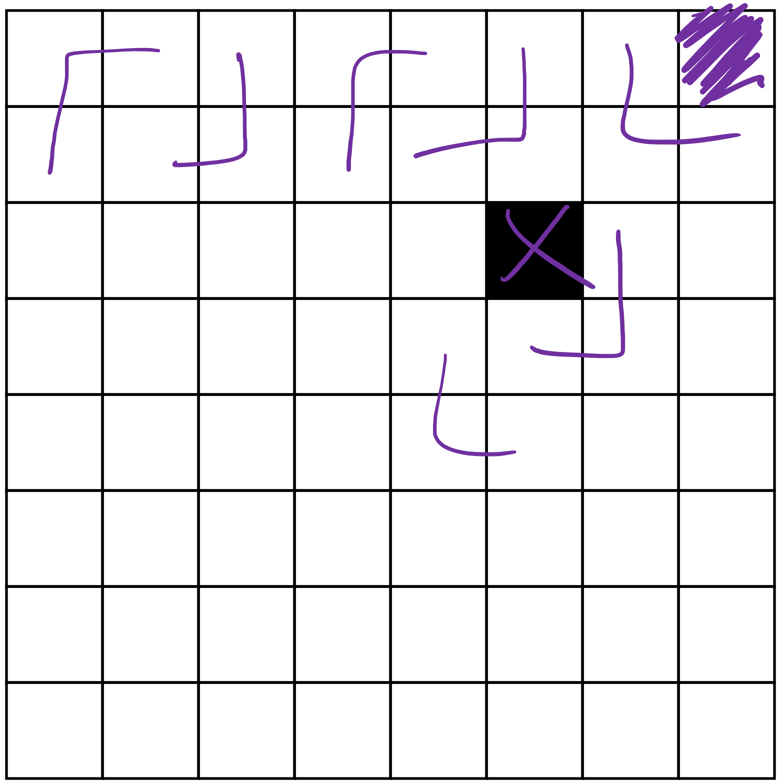
Co-instructors: Robbie Hott and Ray Pettit

Spring 2024

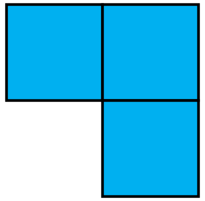
Readings in CLRS 4th edition:

- Section 22.3, Chapter 4, 4.3, 4.4

Warmup



Can you cover an 8×8 grid with 1 square missing using "trominoes?"



Tromino

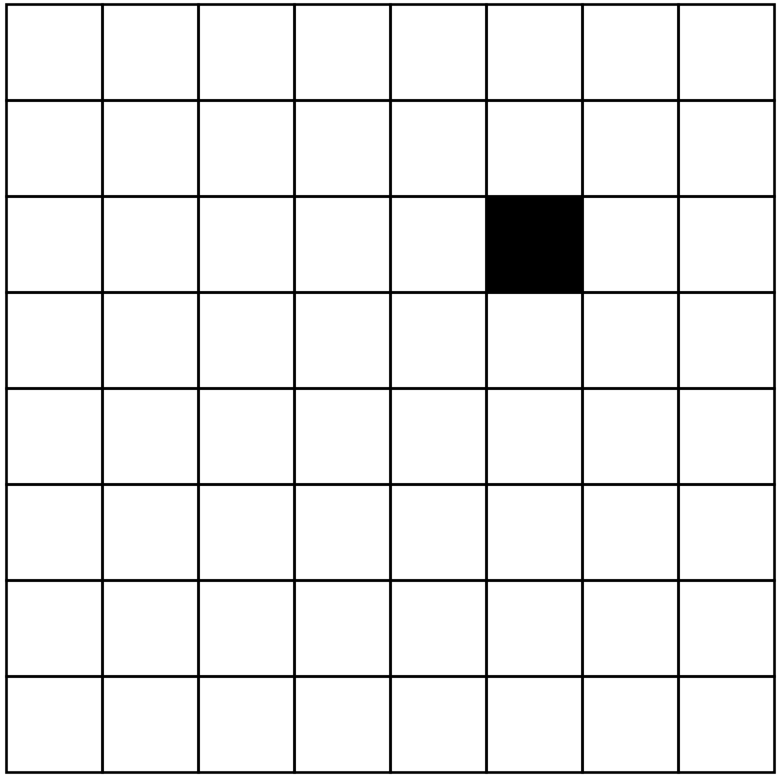
Try it

https://nstarr.people.amherst.edu/trom/puzzle-8by8/

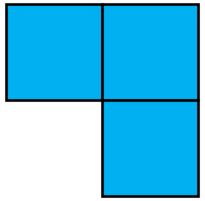
Announcements

- PS3 coming soon!
- PA1 due tomorrow
- Office hours
  - Prof Hott Office Hours: Mondays 11a-12p, Fridays 10-11a and 2-3p
  - Prof Pettit Office Hours: Mondays and Wednesdays 2:30-4:00p
  - TA office hours posted on our website
- Quizzes 1-2 coming February 29, 2024
  - Both quizzes taken the same day
  - If you have SDAC, please schedule for 1 exam (*not a quiz*)

Question



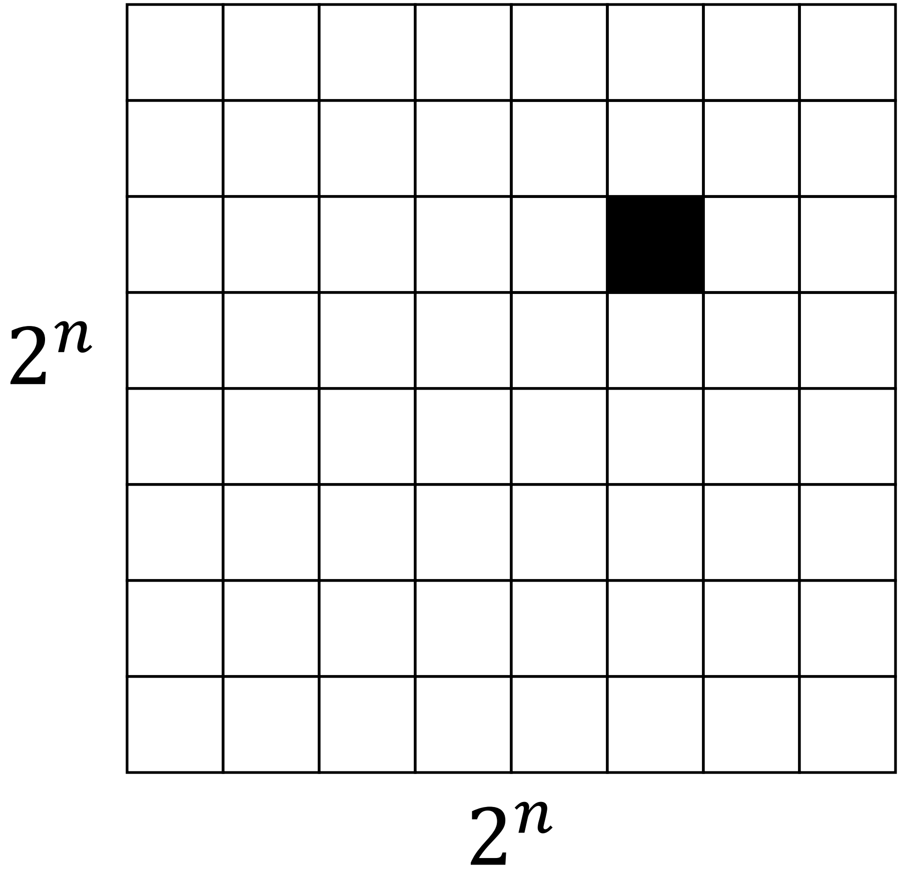
Can you cover an 8×8 grid with 1 square missing using “trominoes?”



Tromino

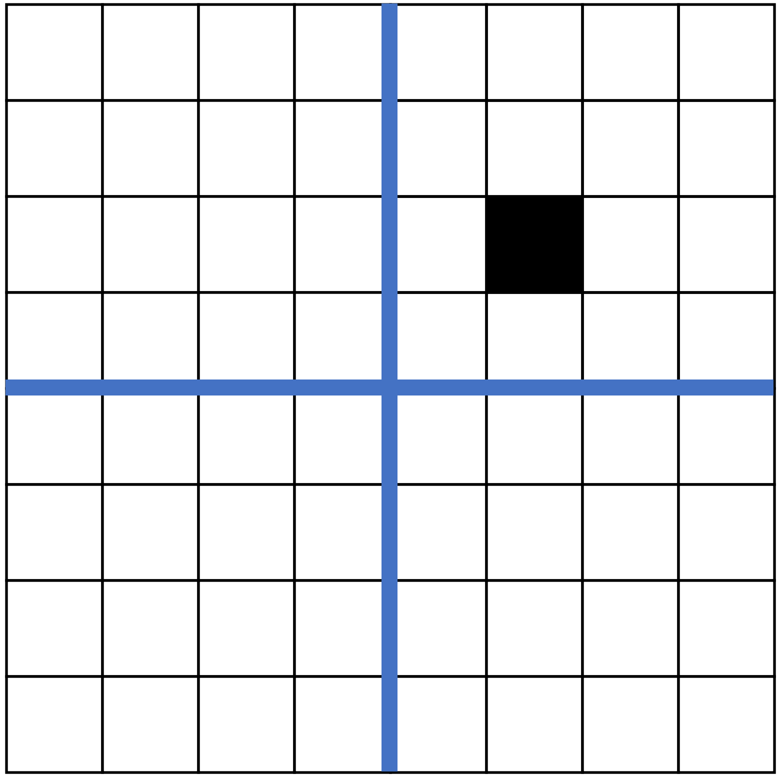
https://nstarr.people.amherst.edu/trom/puzzle-8by8/

Trominoes



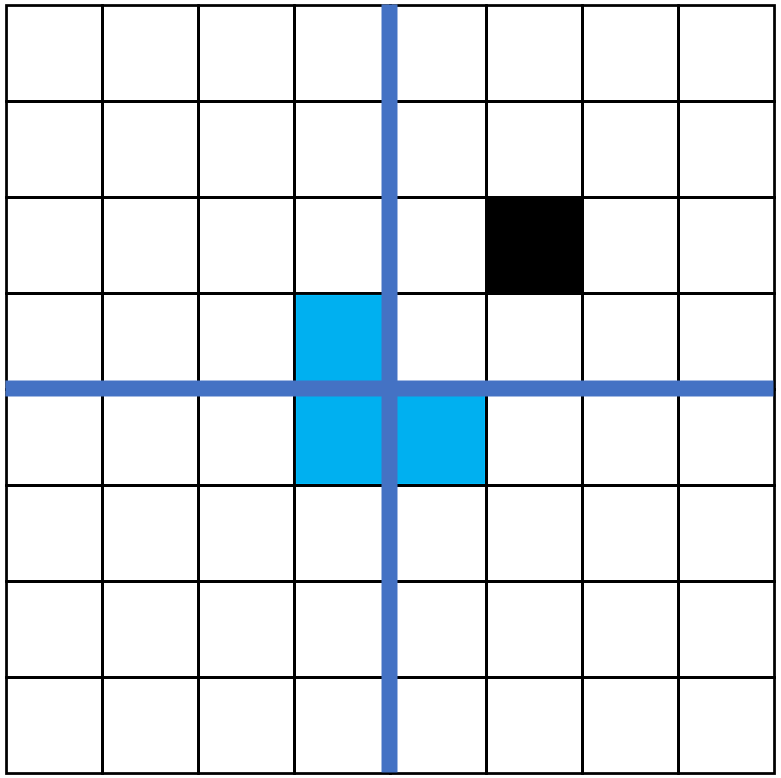
What about larger boards?

Trominoes Puzzle Solution



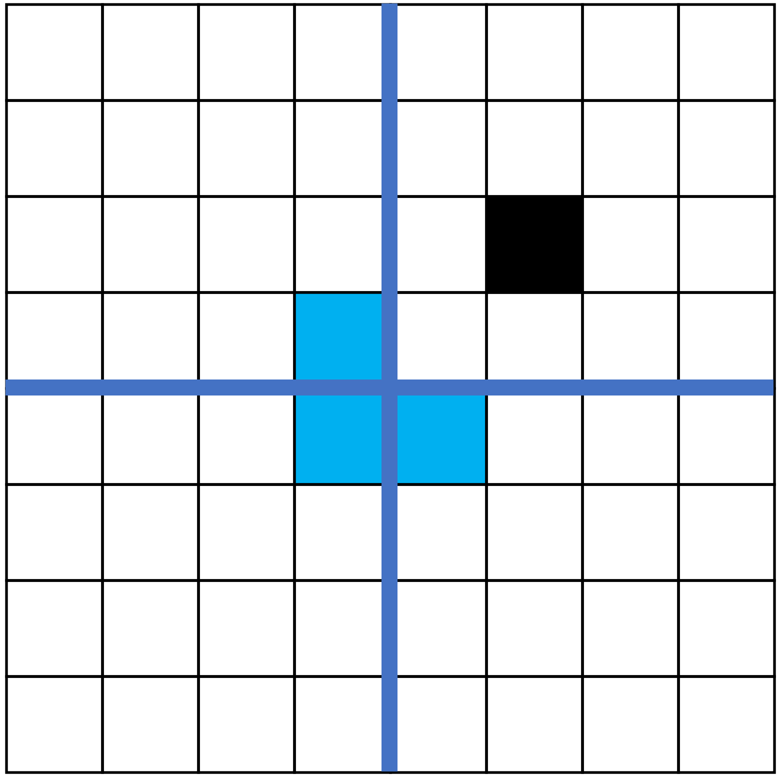
Divide the board into quadrants

Trominoes Puzzle Solution



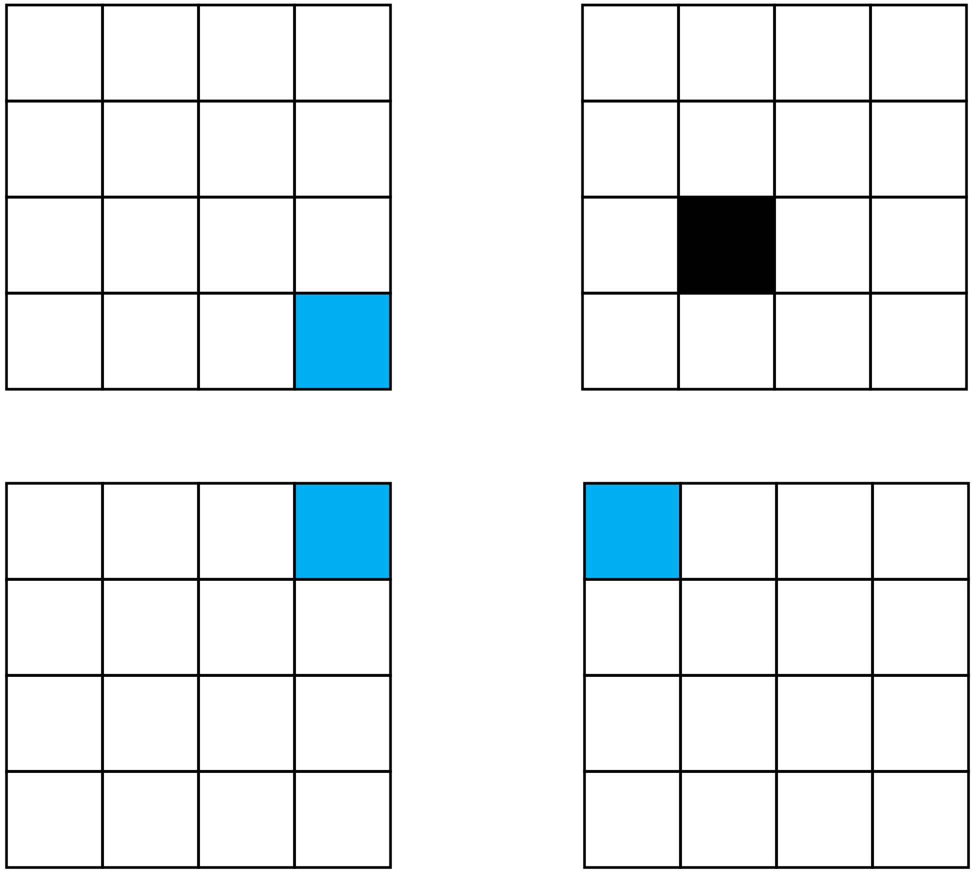
Place a tromino to occupy the three quadrants without the missing piece

Trominoes Puzzle Solution



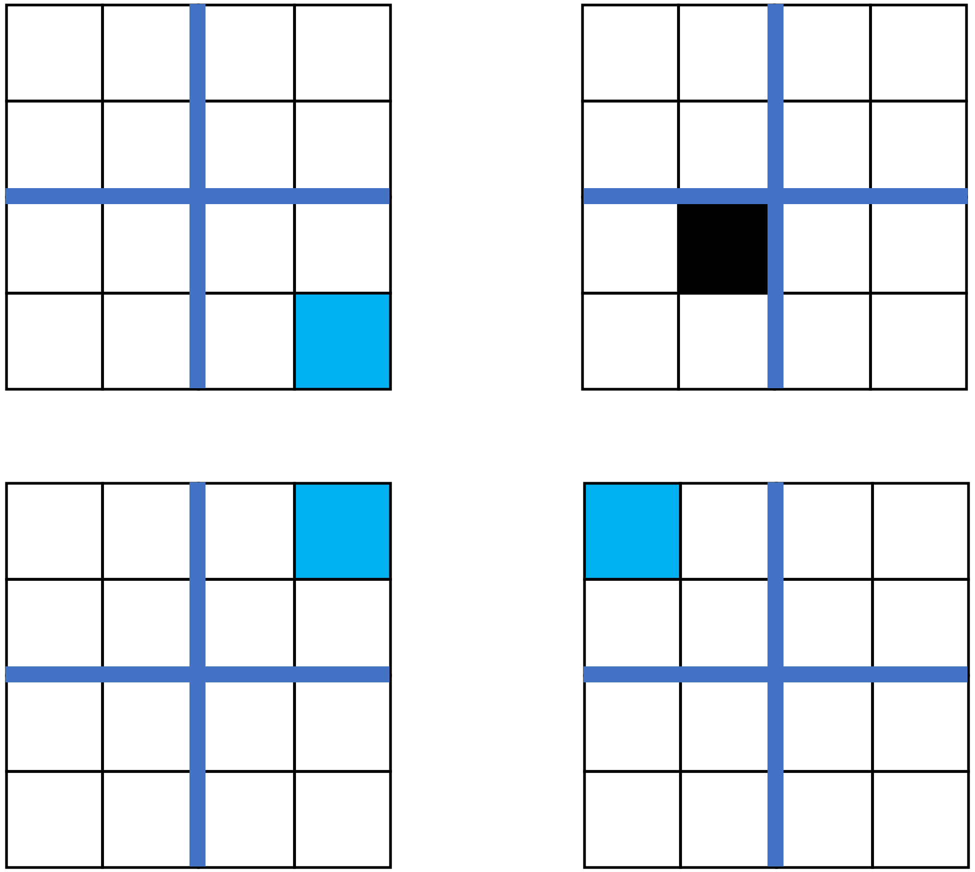
Place a tromino to occupy the three quadrants without the missing piece

Trominoes Puzzle Solution



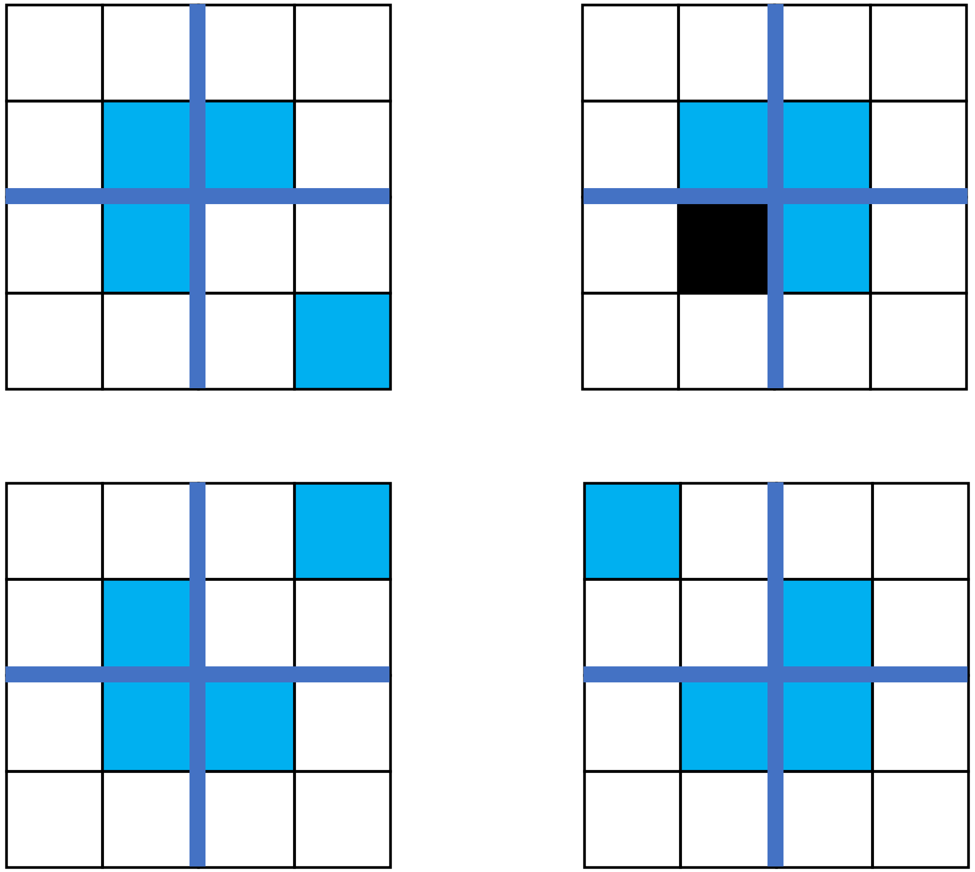
Observe: Each quadrant is now a smaller subproblem!

Trominoes Puzzle Solution



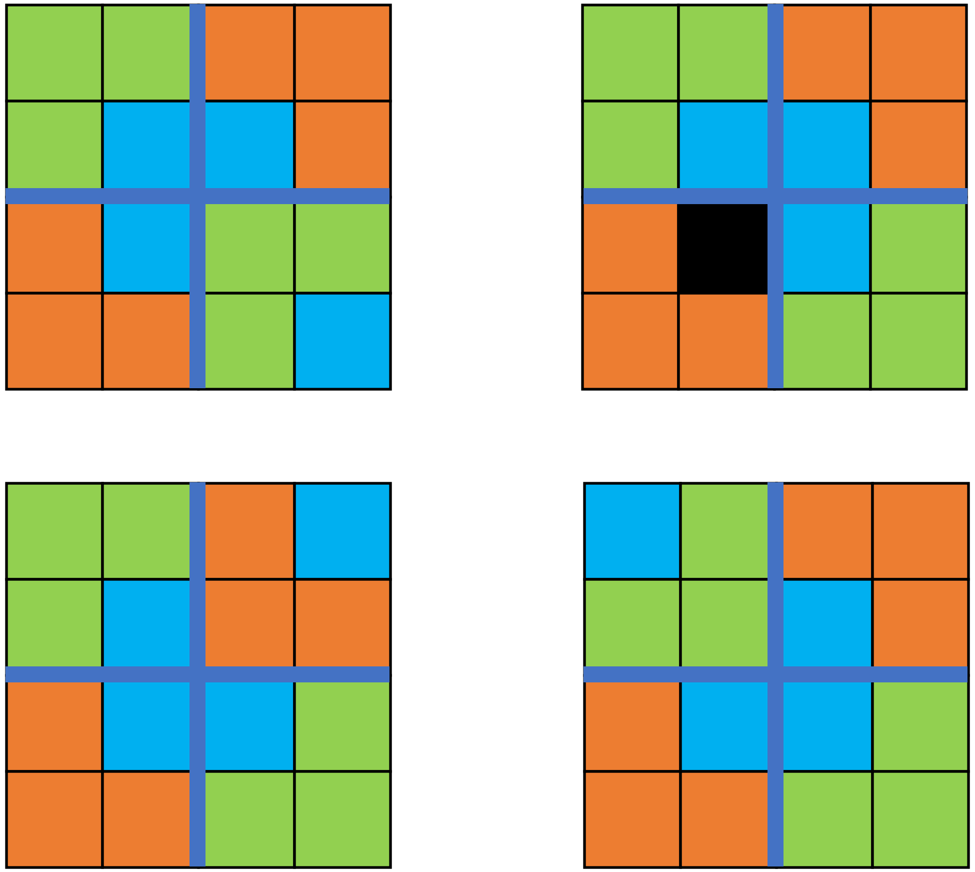
Solve **Recursively**

Trominoes Puzzle Solution



Solve **Recursively**

Trominoes Puzzle Solution



Our first algorithmic technique!

Divide and Conquer

[CLRS Chapter 4]

Divide:

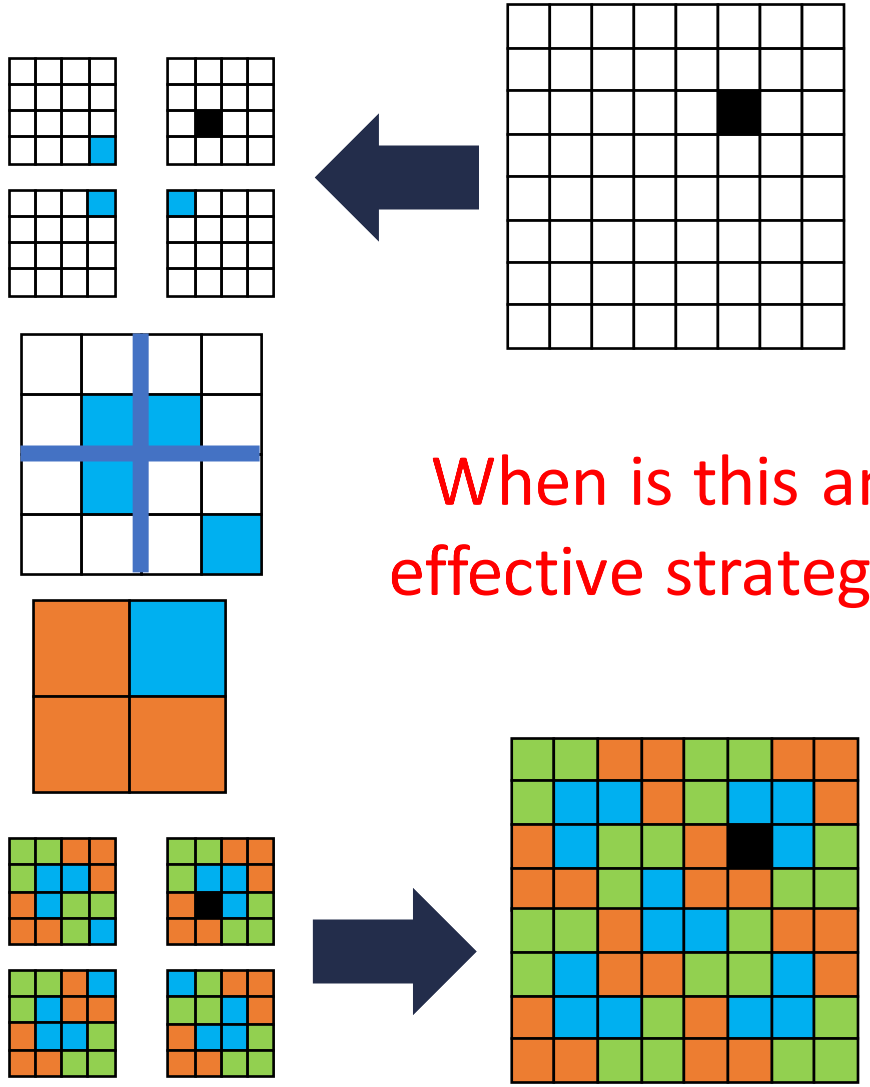
- Break the problem into multiple subproblems, each smaller instances of the original

Conquer:

- If the suproblems are "large":
  - Solve each subproblem recursively
- If the subproblems are "small":
  - Solve them directly (base case)

Combine:

- Merge solutions to subproblems to obtain solution for original problem



When is this an effective strategy?

Remember: 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

MergeSort Divide and Conquer Solution

```
def mergesort(list):
        if list.length < 2:
                return list     #list of size 1 is sorted!
        {listL, listR} = Divide_by_median(list)
        for list in {listL, listR}:
                sortedSubLists.append(mergesort(list))
        solution = merge(sortedL, sortedR)
        return solution
```

Magic

Remember: Merge

Combine: Merge sorted sublists into one sorted list

Inputs:

- 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())

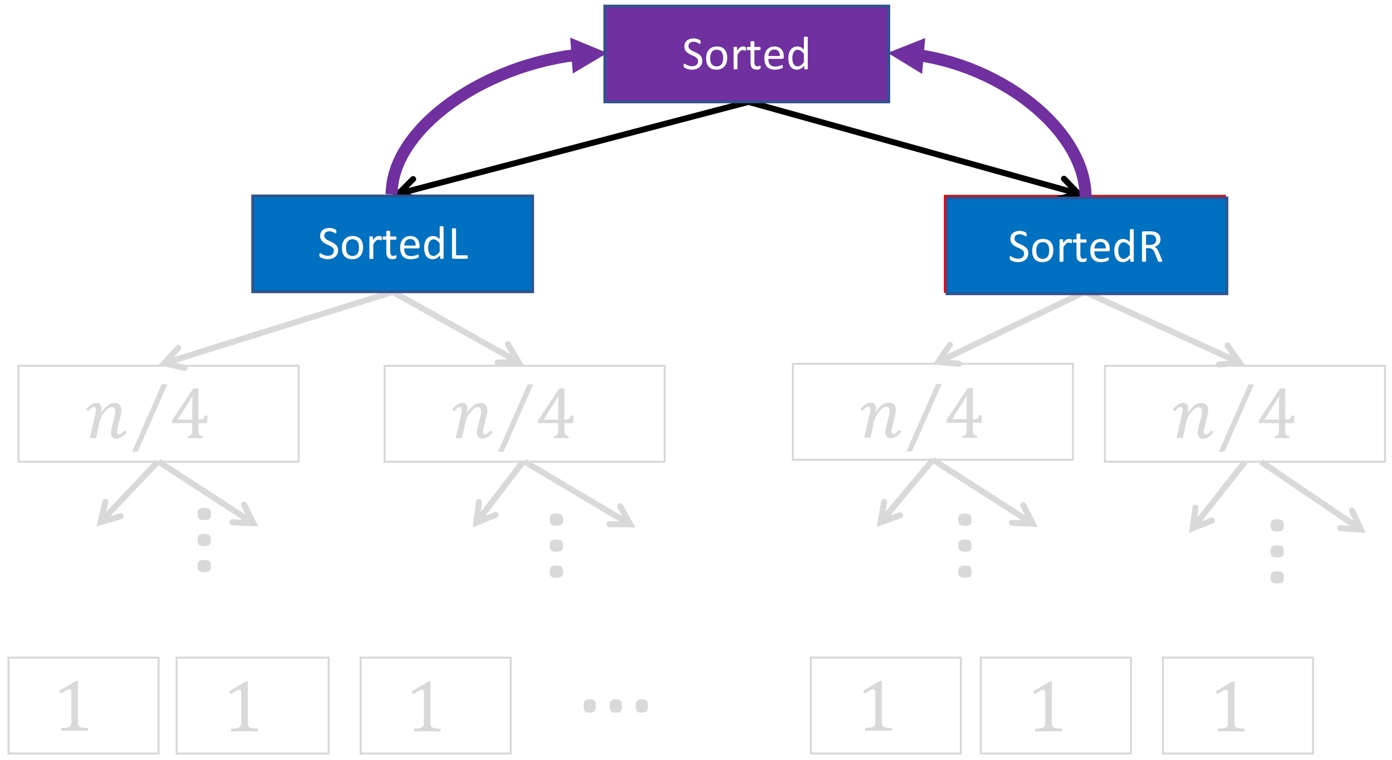
    Else:

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

L_{out}.append(L_1)

L_{out}.append(L_2)

MergeSort Divide and Conquer Solution

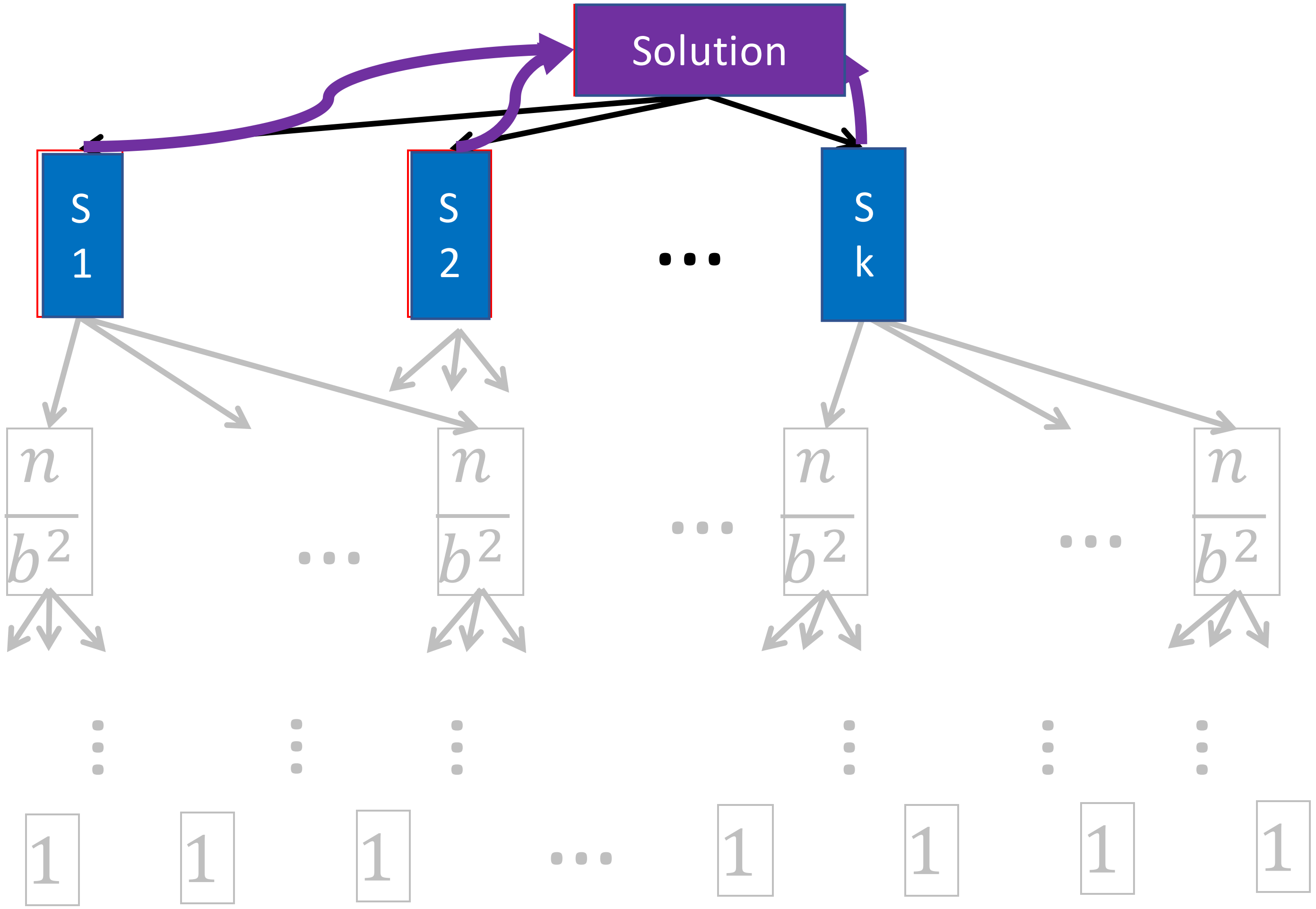


Generic Divide and Conquer Solution

```
def myDCalgo(problem):
        if baseCase(problem):
                solution = solve(problem) #brute force if necessary
                return solution
        subproblems[] = Divide(problem)
        for subproblem in subproblems:
                subsolutions.append(myDCalgo(subproblem))
        solution = Combine(subsolutions)
        return solution
```

Magic!

Generic Divide and Conquer Solution



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 smaller problems of size $s_1, \ldots, s_k$

Combine: $C(n)$ time

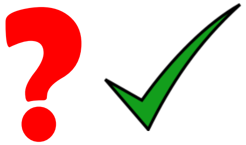
Recurrence:

- $T(n) = D(n) + \sum_{i \in [k]} T(s_i) + C(n)$

Recurrence Solving Techniques

Tree

get a picture of recursion



Guess/Check

guess and use induction to prove

"Cookbook"

MAGIC!

Substitution

substitute in to simplify

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 $\frac{n}{2}$

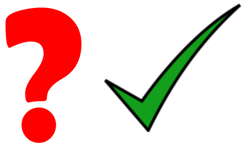
Combine: n comparisons

Recurrence:

- $T(n) = 2T(n/2) + n$

Recurrence Solving Techniques

Tree



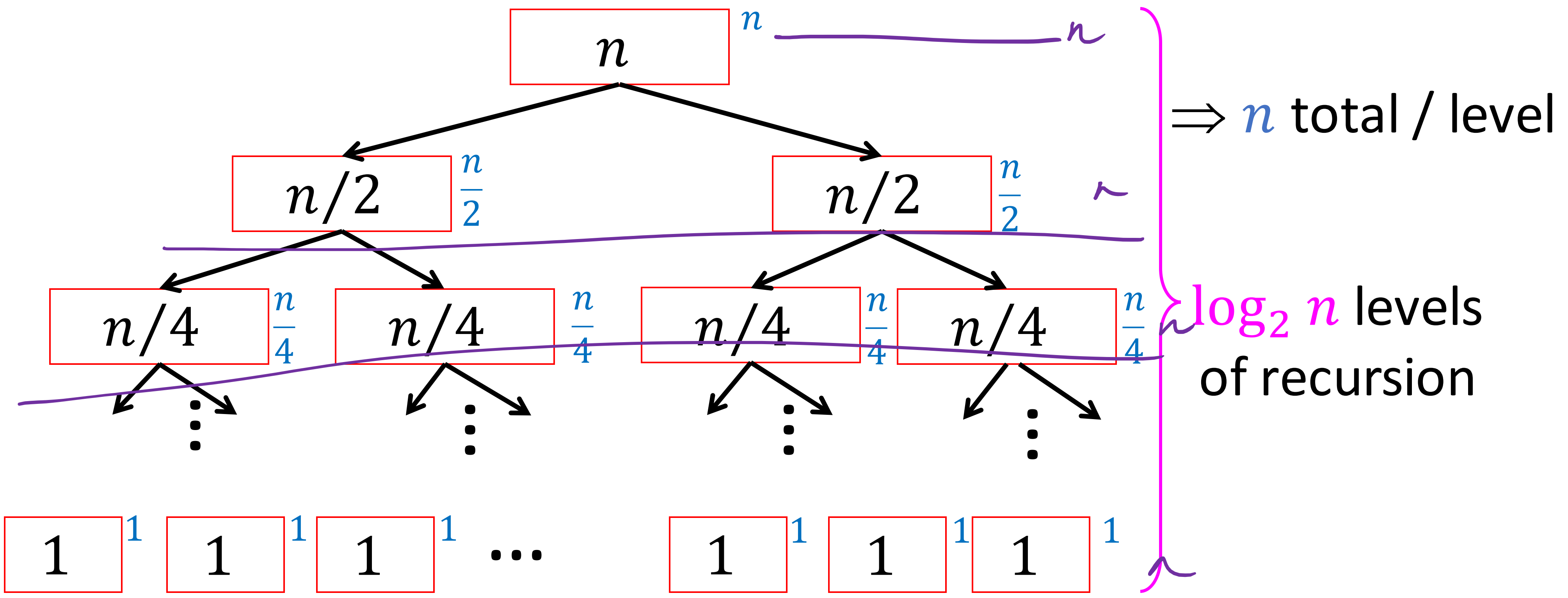
Guess/Check

"Cookbook"

Substitution

Tree method

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



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

An aside... another warm up?

Warm up

Simplify:

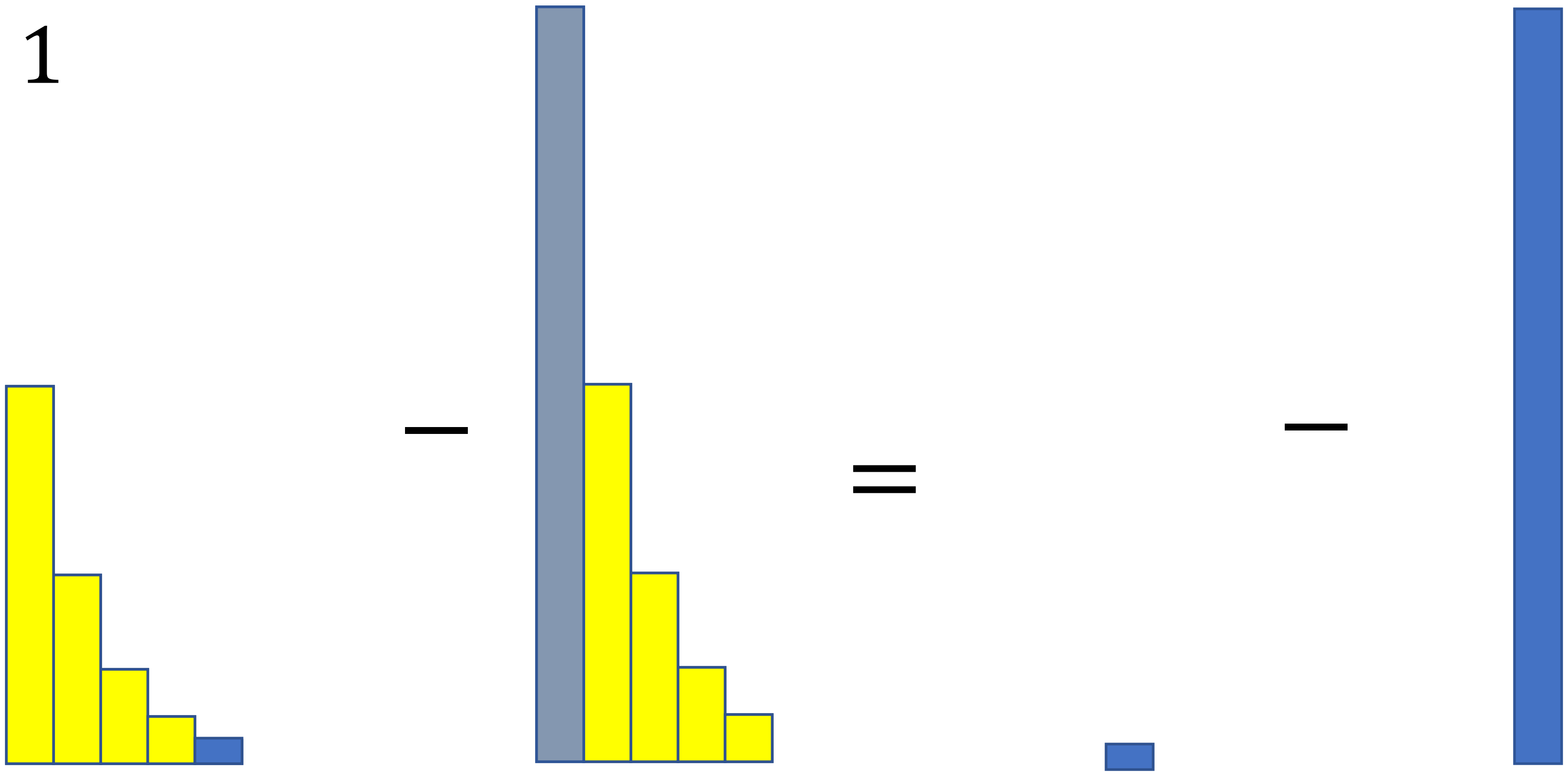
$$(1 + a + a^2 + a^3 + a^4 + \cdots + a^L)(a - 1) = ?$$

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

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

Finite Geometric Series

$a < 1$



The series multiplied by a

$(1 + a + a^2 + \cdots + a^L)a$

The series

$(1 + a + a^2 + \cdots + a^L)1$

The next term in the series

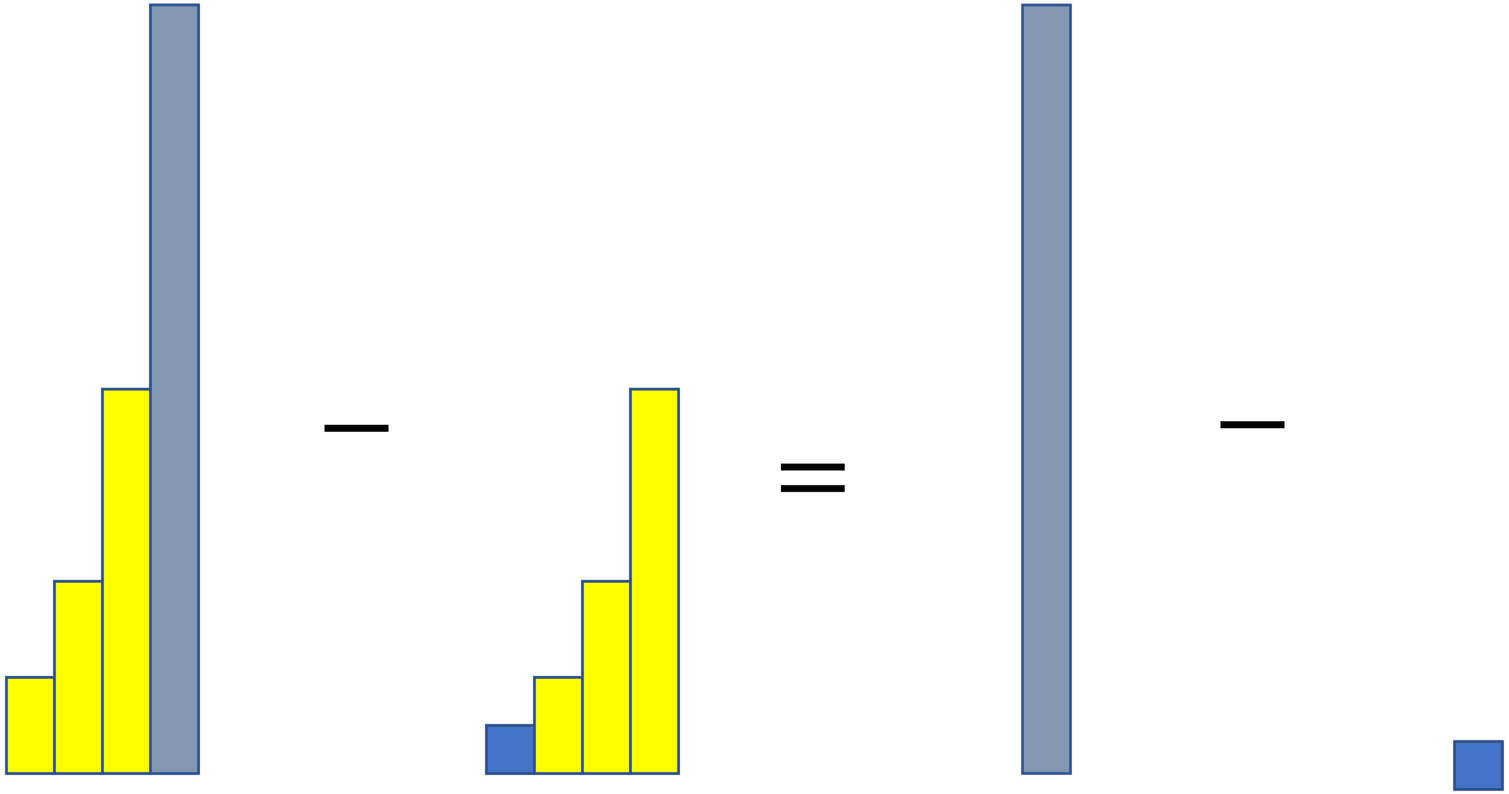
a^{L+1}

The first term

1

Finite Geometric Series

$a > 1$



The series multiplied by a

$(1 + a + a^2 + \cdots + a^L)a$

−

The series

$(1 + a + a^2 + \cdots + a^L)1$

=

The next term in the series

a^{L+1}

−

The first term

1

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

How do we measure input size?

number of digits

What do we "count" for run time?

number of elementary operations
(single-digit multiplications)

"Schoolbook" Multiplication

How many multiplications?

$$
\begin{array}{r}
4102 \\
\times\ 1819 \\
\hline
36918 \\
41020 \\
3281600 \\
+\ 4102000 \\
\hline
7461538
\end{array}
$$

n-digit numbers

n mults

n mults

n mults

n mults

n levels $\Rightarrow \Theta(n^2)$

What about cost of additions?

$\Theta(n^2)$

“Schoolbook” Multiplication

Can we do better?

How many multiplications?

$$
\begin{array}{r}
4102 \\
\times\ 1819 \\
\hline
36918 \\
4102 \\
32816 \\
+\ 4102 \\
\hline
7461538
\end{array}
$$

n-digit numbers

n mults

n mults

n mults

n mults

n levels $\Rightarrow \Theta(n^2)$

What about cost of additions?

$\Theta(n^2)$

Divide and Conquer Multiplication

1. Break into smaller subproblems

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

$$= \; 10^{n}(a \times c) +$$

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

$$(b \times d)$$

a = 41
b = 02

4100
+ 02
4102

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 $n = 1$: (i.e. one digit each)
  - Compute ac, ad, bc, bd directly (base case)

Combine:

- $10^n(ac) + 10^{n/2}(ad + bc) + bd$

For simplicity, assume that $n = 2^k$ is a power of 2

Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

$$10^n(ac) + 10^{n/2}(ad + bc) + bd$$

Recursively solve

$$T(n)$$

Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

$$10^n(ac) + 10^{n/2}(ad + bc) + bd$$

Recursively solve

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

Need to compute 4 multiplications, each of size $n/2$

Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

$$10^n(ac) + 10^{n/2}(ad + bc) + bd$$

Recursively solve

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

Need to compute 4 multiplications, each of size $n/2$

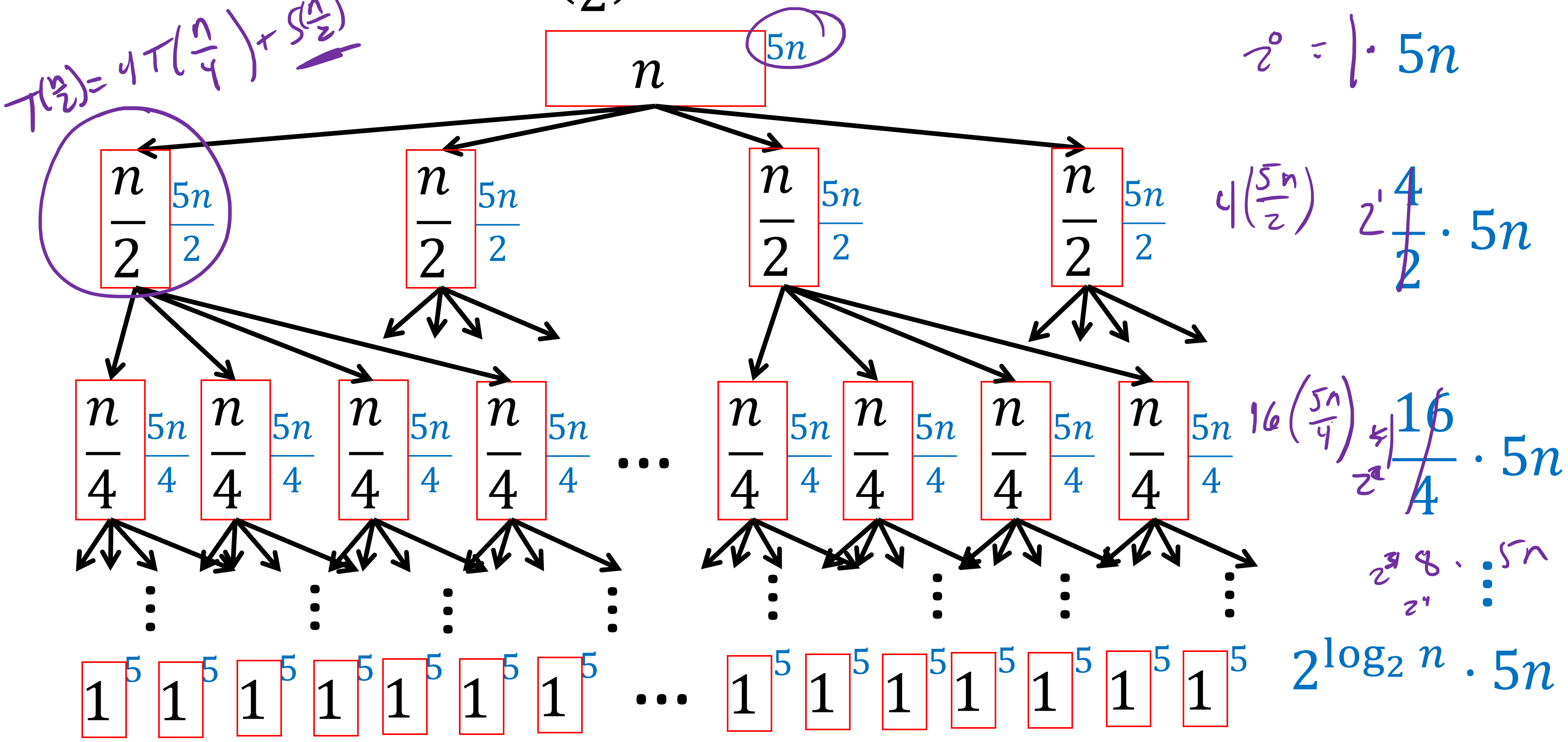
2 shifts and 3 additions on n-bit values

Divide and Conquer Multiplication

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 Multiplication

3. Use asymptotic notation to simplify

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

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

$$2^{\log_2 n + 1} = 2^{\log_2 n} \cdot 2 = n \cdot 2$$

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

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

Karatsuba Multiplication

1. Break into smaller subproblems

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

$$= 10^{n}(a \times c) +$$

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

$$(b \times d)$$

Recall: previous divide-and-conquer recursively computed ac, ad, bc, bd

Karatsuba Multiplication

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

Can't avoid these

This can be simplified!

	a	b
×	c	d

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

$$ac + ad + bc + bd$$

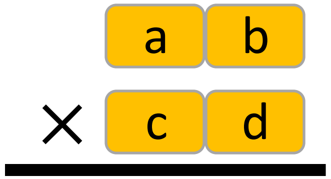
$$ad + bc = (a + b)(c + d) - ac - bd$$

Two multiplications

One multiplication

Karatsuba Multiplication

2. Use recurrence relation to express recursive running time



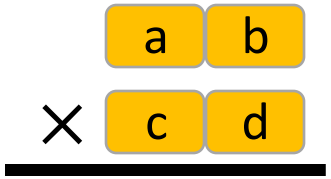
$$10^n(ac) + 10^{n/2}((a+b)(c+d) - ac - bd) + bd$$

Recursively solve

$$T(n) =$$

Karatsuba Multiplication

2. Use recurrence relation to express recursive running time



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

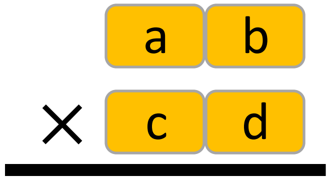
Recursively solve

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

Need to compute **3** multiplications, each of size $n/2$: ac, bd, $(a+b)(b+c)$

Karatsuba Multiplication

2. Use recurrence relation to express recursive running time



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

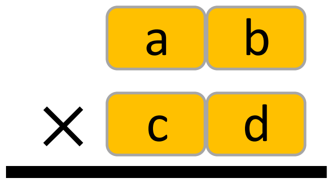
Recursively solve

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

Need to compute **3** multiplications, each of size $n/2$: ac, bd, $(a+b)(c+d)$

2 shifts and 6 additions on n-bit values

Karatsuba Multiplication



Divide:

- Break n-digit numbers into four numbers of $n/2$ digits each (call them a, b, c, d)

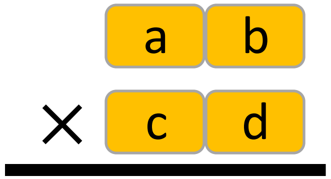
Conquer:

- If $n > 1$:
  - Recursively compute ac, bd, $(a+b)(c+d)$
- If $n = 1$:
  - Compute ac, bd, $(a+b)(c+d)$ directly (base case)

Combine:

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

Karatsuba Multiplication



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

<u>Pseudocode:</u>

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

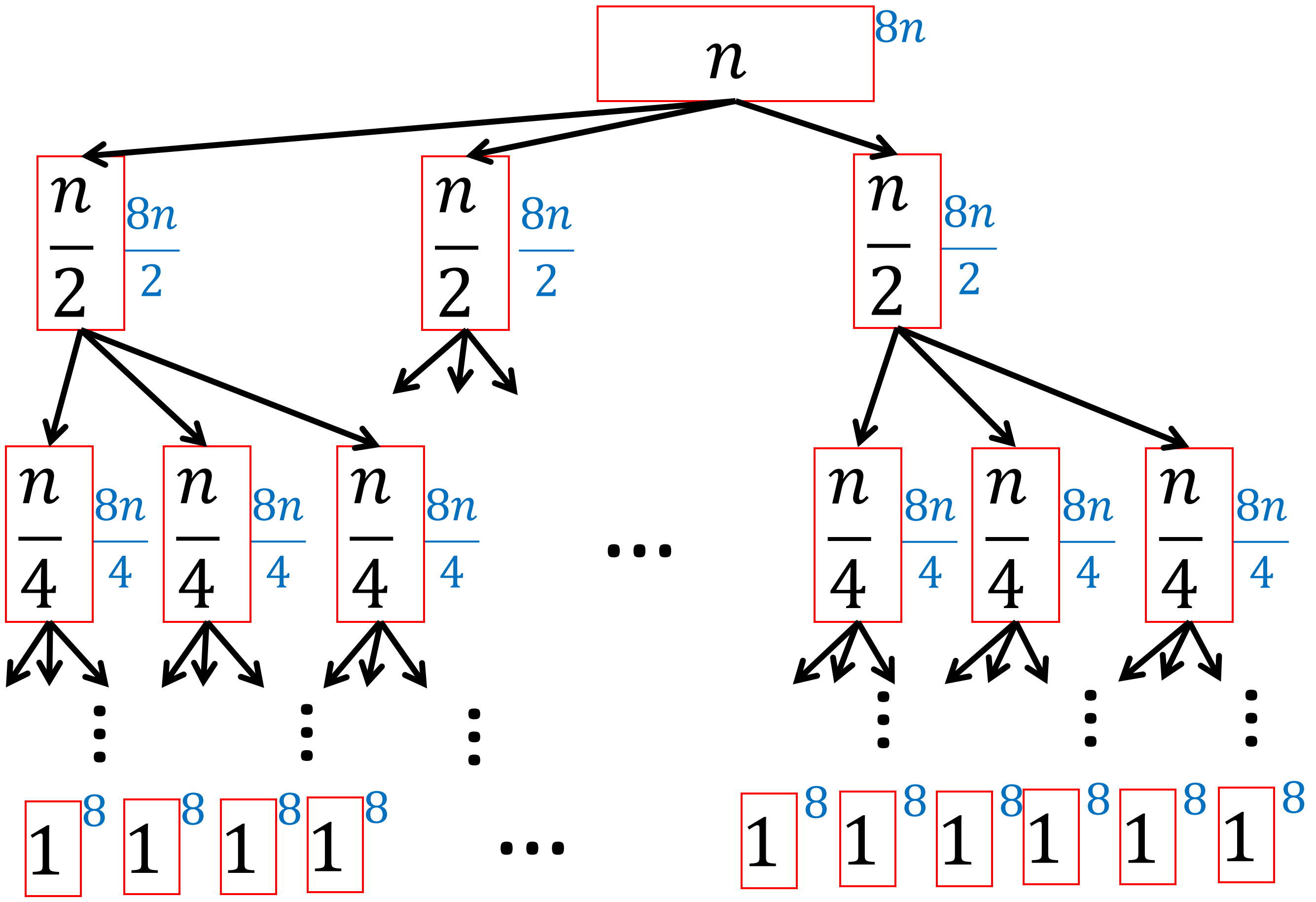
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Karatsuba

3. Use asymptotic notation to simplify

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

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



$$8n \cdot 1$$

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

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

$$\vdots$$

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

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 supplement)

Karatsuba

Karatsuba

Karatsuba Multiplication

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

How to simplify this
(using asymptotic notation)?

Drop constant multiples

Karatsuba Multiplication

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

How to simplify this (using asymptotic notation)?

$$= \Theta\left(n\left(\left(3/2\right)^{\log_2 n+1} - 1\right)\right)$$

Drop constant multiples

$$= \Theta\left(\frac{3}{2}n \cdot \left(3/2\right)^{\log_2 n} - n\right)$$

Distribute terms

Karatsuba Multiplication

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

How to simplify this (using asymptotic notation)?

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

Drop constant multiples

$$= \Theta\left(\frac{3}{2}n \cdot \left({}^{3}/_{2}\right)^{\log_2 n} - n\right)$$

Distribute terms

$$= \Theta\left(n \cdot \left({}^{3}/_{2}\right)^{\log_2 n}\right)$$

Drop constants and low-order terms

Karatsuba Multiplication

$$T(n) = \Theta\left(n \cdot \left({}^{3}/_{2}\right)^{\log_2 n}\right)$$

How to simplify this (using asymptotic notation)?

Properties of logarithms:

$$2^{\log_2 n} = n$$

$$3^{\log_2 n} = 2^{\log_2\left(3^{\log_2 n}\right)} = 2^{(\log_2 n)(\log_2 3)} = \left(2^{\log_2 n}\right)^{\log_2 3} = n^{\log_2 3}$$

$a^{bc} = \left(a^b\right)^c$

$2^{\log_2 n} = n$

$\log a^b = b \log a$

$2^{\log_2 n} = n$

Karatsuba Multiplication

$$T(n) = \Theta\left(n \cdot \left({}^{3}/_{2}\right)^{\log_2 n}\right)$$

$$= \Theta\left(n \cdot \left(\frac{3^{\log_2 n}}{2^{\log_2 n}}\right)\right)$$

$$= \Theta\left(n \cdot \left(\frac{n^{\log_2 3}}{n}\right)\right)$$

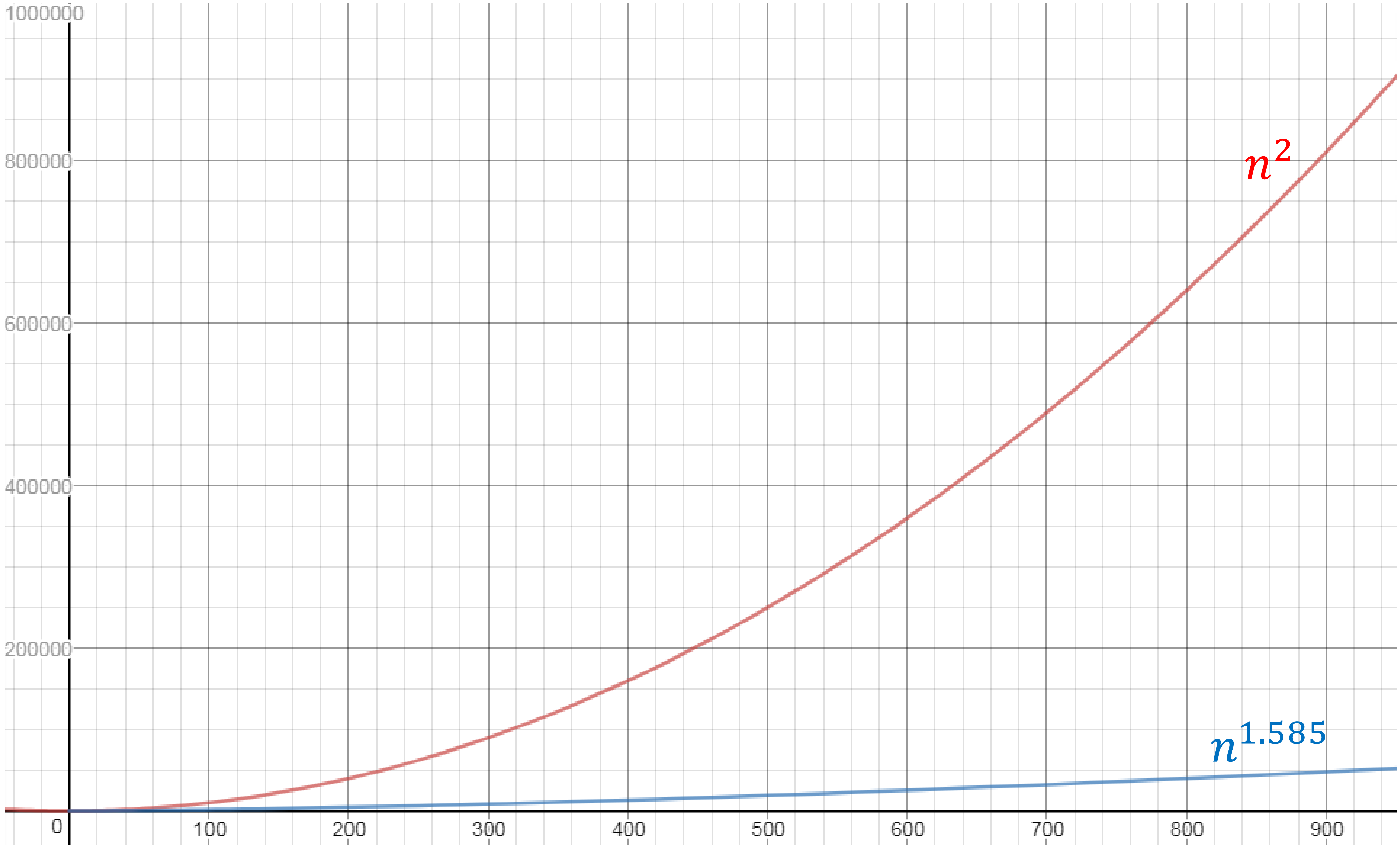
$$= \Theta\left(n^{\log_2 3}\right) \approx \Theta\left(n^{1.585}\right)$$

How to simplify this (using asymptotic notation)?

$$2^{\log_2 n} = n$$

$$3^{\log_2 n} = n^{\log_2 3}$$

Strictly better than schoolbook method!


1000000
800000
600000
400000
200000
0
100
200
300
400
500
600
700
800
900
n^2
$n^{1.585}$