CS 4102: Algorithms
Spring 2020

Lecture 3: Divide and Conquer

Co-instructors: Robbie Hott and Tom Horton

(These are slides for Horton's section)

Warm up

Simplify:

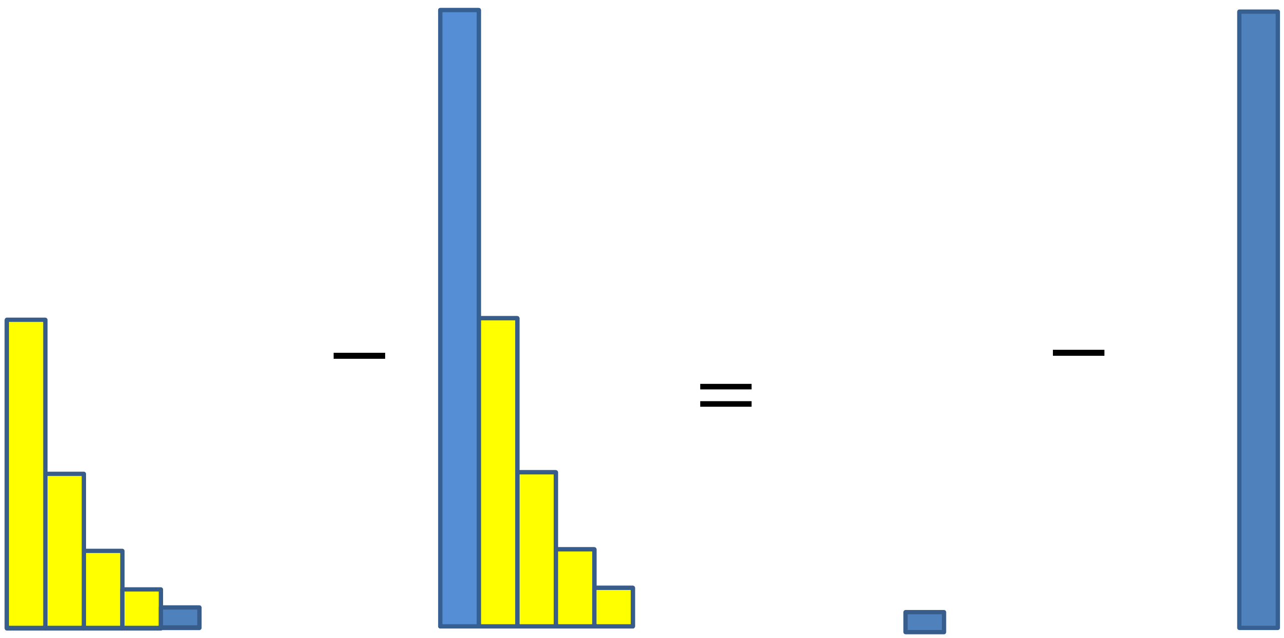
$$(1 + a + a^2 + a^3 + a^4 + \cdots + a^L)(a - 1) = ?$$

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

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

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

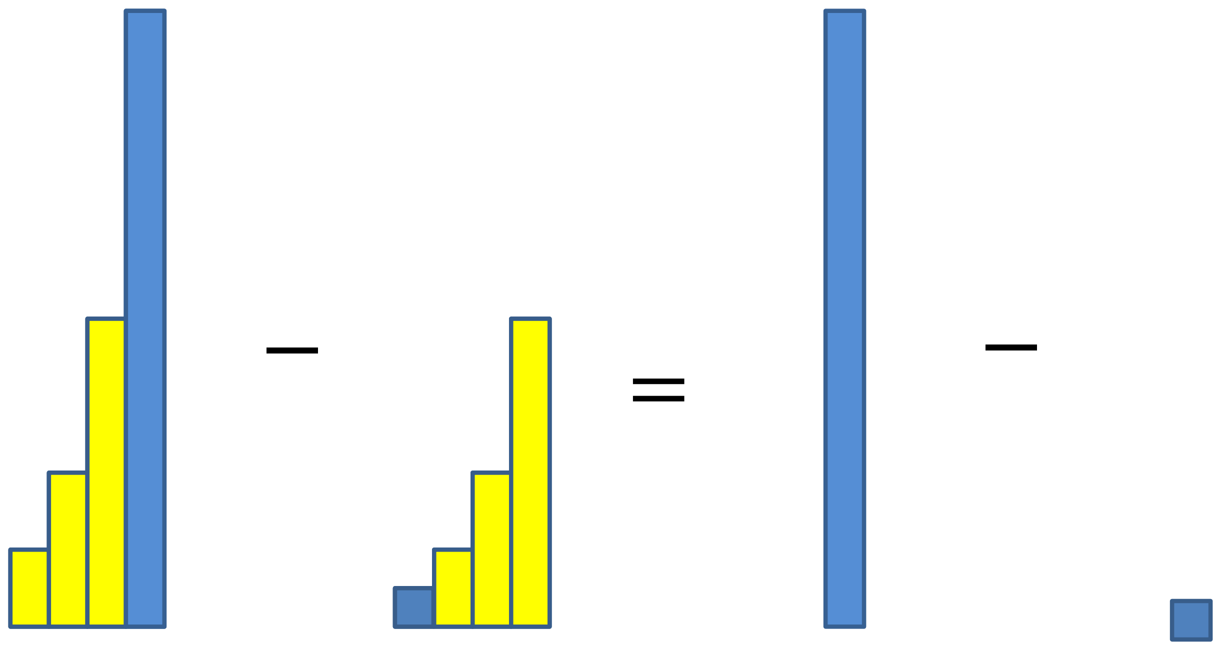
a^{L+1}

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

Today’s Keywords

- Divide and Conquer
- Recurrences
- Merge Sort
- Karatsuba
- Tree Method

CLRS Readings

- Chapter 4

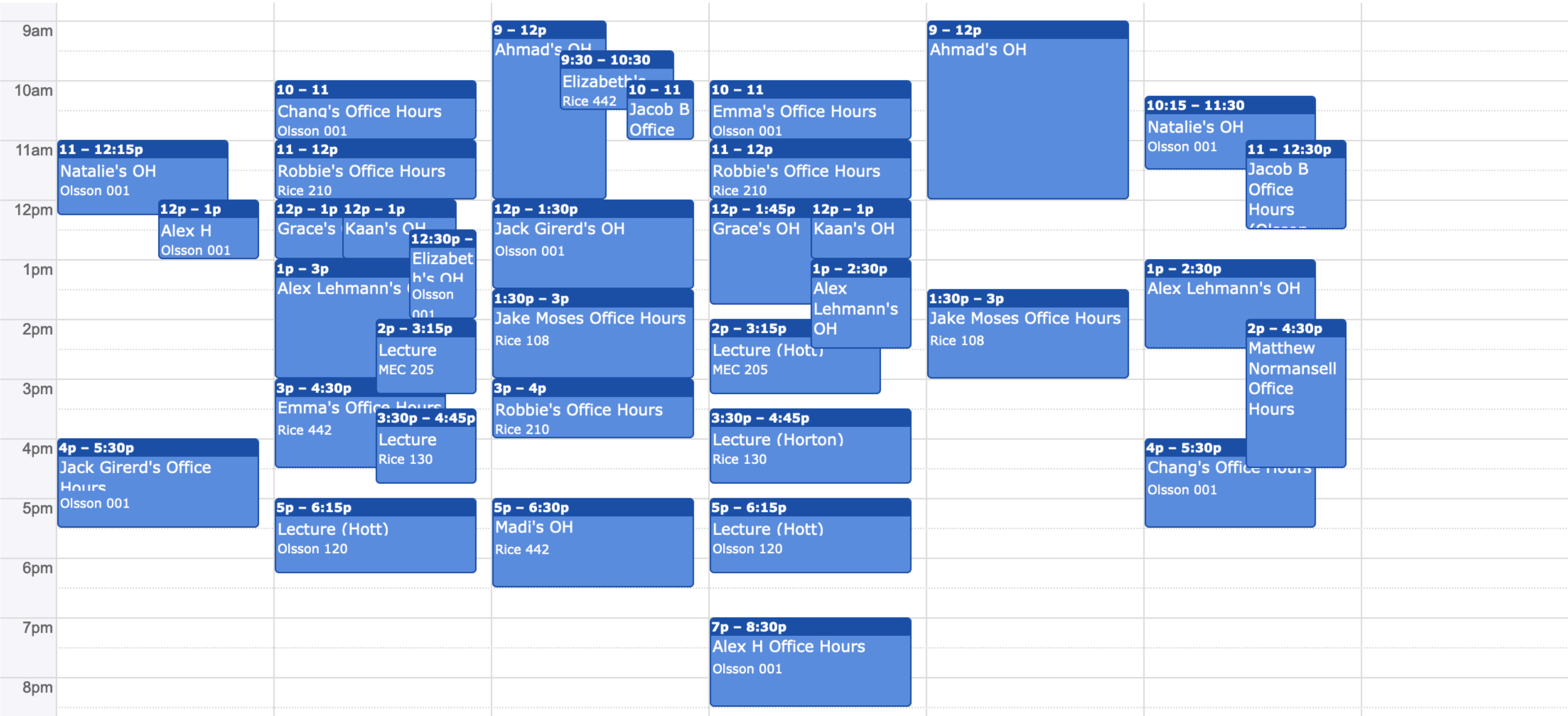
Homeworks

- HW1 due Thursday, January 30 at 11pm
  - Start early!
  - Written (use Latex!) – Submit BOTH pdf and zip!
  - Asymptotic notation
  - Recurrences
  - Divide and Conquer

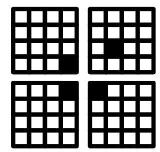
Homework Help Algorithm

- Algorithm: How to ask a question about homework (efficiently)
  1. Check to see if your question is already on piazza
  2. If it's not on piazza, ask on piazza
  3. Look for other questions you know the answer to, and provide answers to any that you see
  4. TA office hours
  5. Instructor office hours
  6. Email, set up a meeting

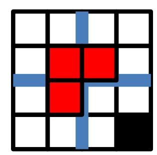
Office Hours

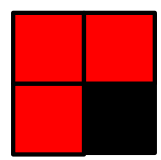


Divide and Conquer*

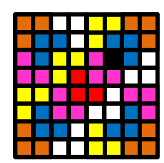


- **Divide**:
  - Break the problem into multiple subproblems, each smaller instances of the original
- **Conquer**:
  - If the subproblems are "large":
    - Solve each subproblem recursively
  - If the subproblems are "small":
    - Solve them directly (base case)





- **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**: $\mathrm{C}(n)$ time
- **Recurrence:**
  - $T(n) = D(n) + \sum T(s) + C(n)$

Recurrence Solving Techniques

Tree

get a picture of recursion

Guess/Check

guess and use induction to prove

“Cookbook” MAGIC!

Substitution

substitute in to simplify

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)

$O(n)$

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 subproblems, size $\frac{n}{2}$
- **Combine**: n comparisons
- **Recurrence:**

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

Recurrence Solving Techniques

Tree

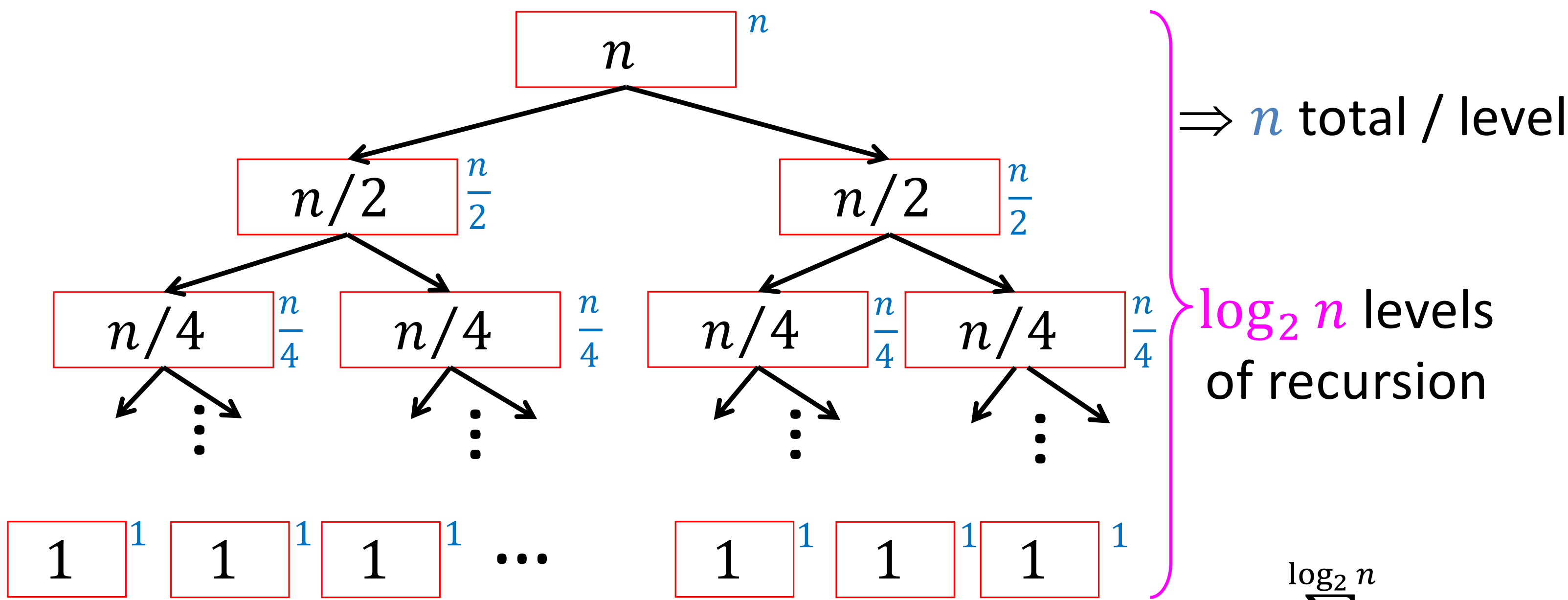
Guess/Check

"Cookbook"

Substitution

Tree method

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



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

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

Can we do better?

How many total multiplications?

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

n-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

Divide and Conquer method

1. Break into smaller subproblems

$$\boxed{a}\,\boxed{b} = 10^{\frac{n}{2}}\boxed{a} + \boxed{b}$$

$$\times\ \boxed{c}\,\boxed{d} = 10^{\frac{n}{2}}\boxed{c} + \boxed{d}$$

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

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

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

Remember "FOIL"?

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**:

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

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

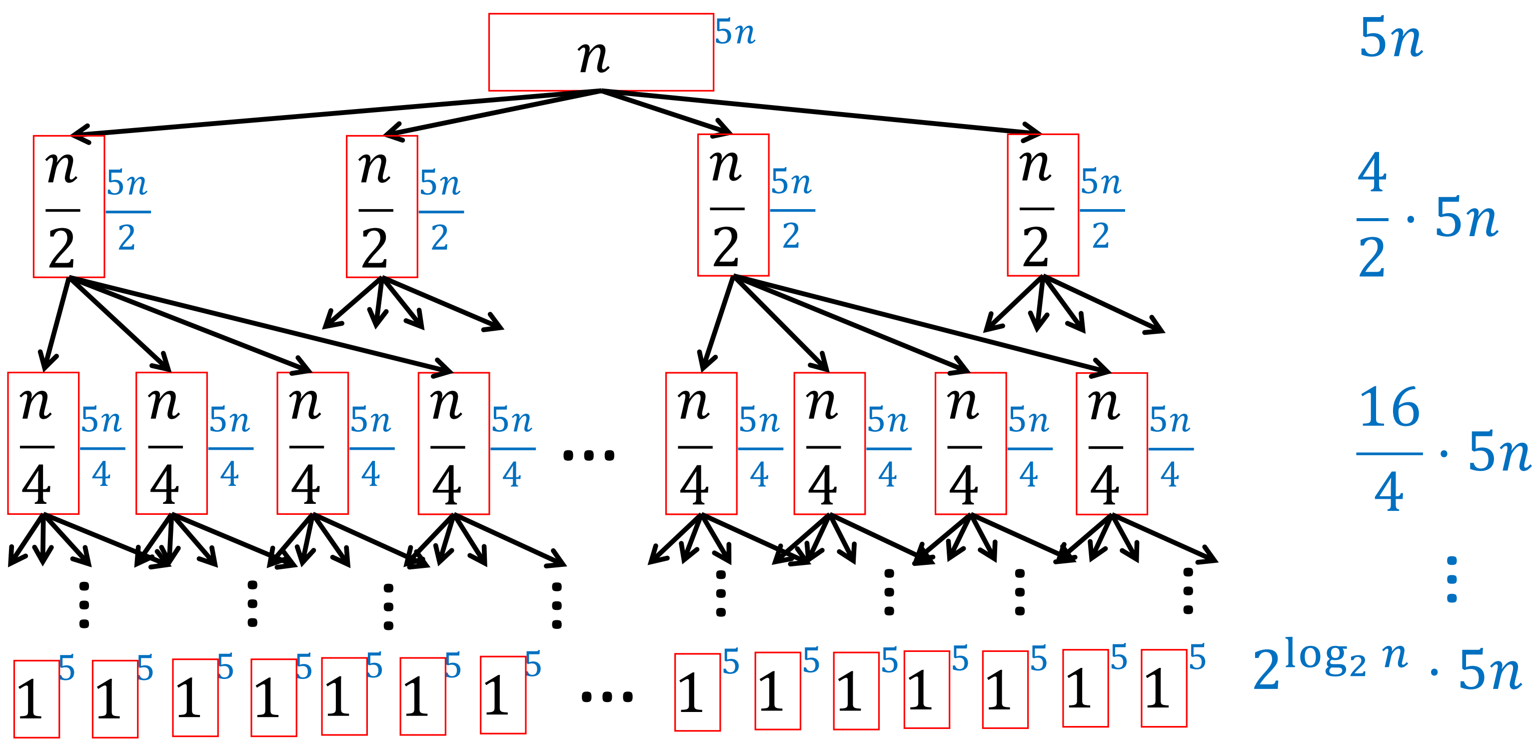
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Divide and Conquer method

3. Use asymptotic notation to simplify

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

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



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

$$\boxed{a}\boxed{b} = 10^{\frac{n}{2}}\boxed{a} + \boxed{b}$$

$$\times\ \boxed{c}\boxed{d} = 10^{\frac{n}{2}}\boxed{c} + \boxed{d}$$

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

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

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

Karatsuba

$$\begin{array}{rcc} & a & b \\ \times & c & d \\ \hline \end{array}$$

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

Can't avoid these

This can be re-written
in terms of the others

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

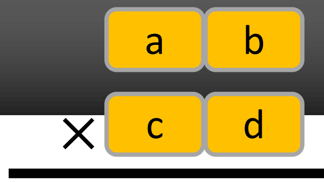
$$\boxed{ac} + \boxed{ad + bc} + \boxed{bd}$$

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

Two
multiplications

One "new" multiplication

Karatsuba



2\. Use recurrence relation to express recursive running time

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

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

$$\begin{array}{rcc} & a & b \\ \times & c & d \\ \hline \end{array}$$

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

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

<u>Pseudocode</u>

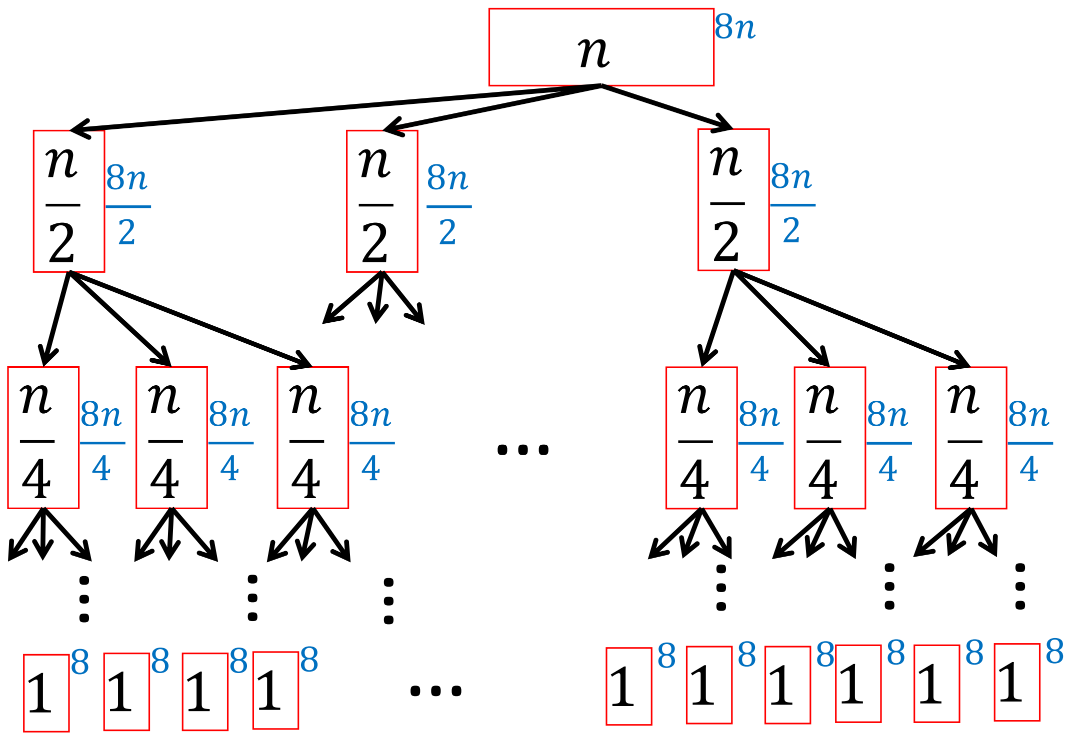
1. $x \leftarrow$ Karatsuba(a, c)
2. $y \leftarrow$ Karatsuba(b, d)
3. $z \leftarrow$ Karatsuba$(a+b, c+d) - x - y$
4. Return $10^n x + 10^{n/2} z + y$

Karatsuba

3. Use asymptotic notation to simplify

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

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



$$8n \cdot 1$$

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

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

$$\vdots$$

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

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

3. Use asymptotic notation to simplify

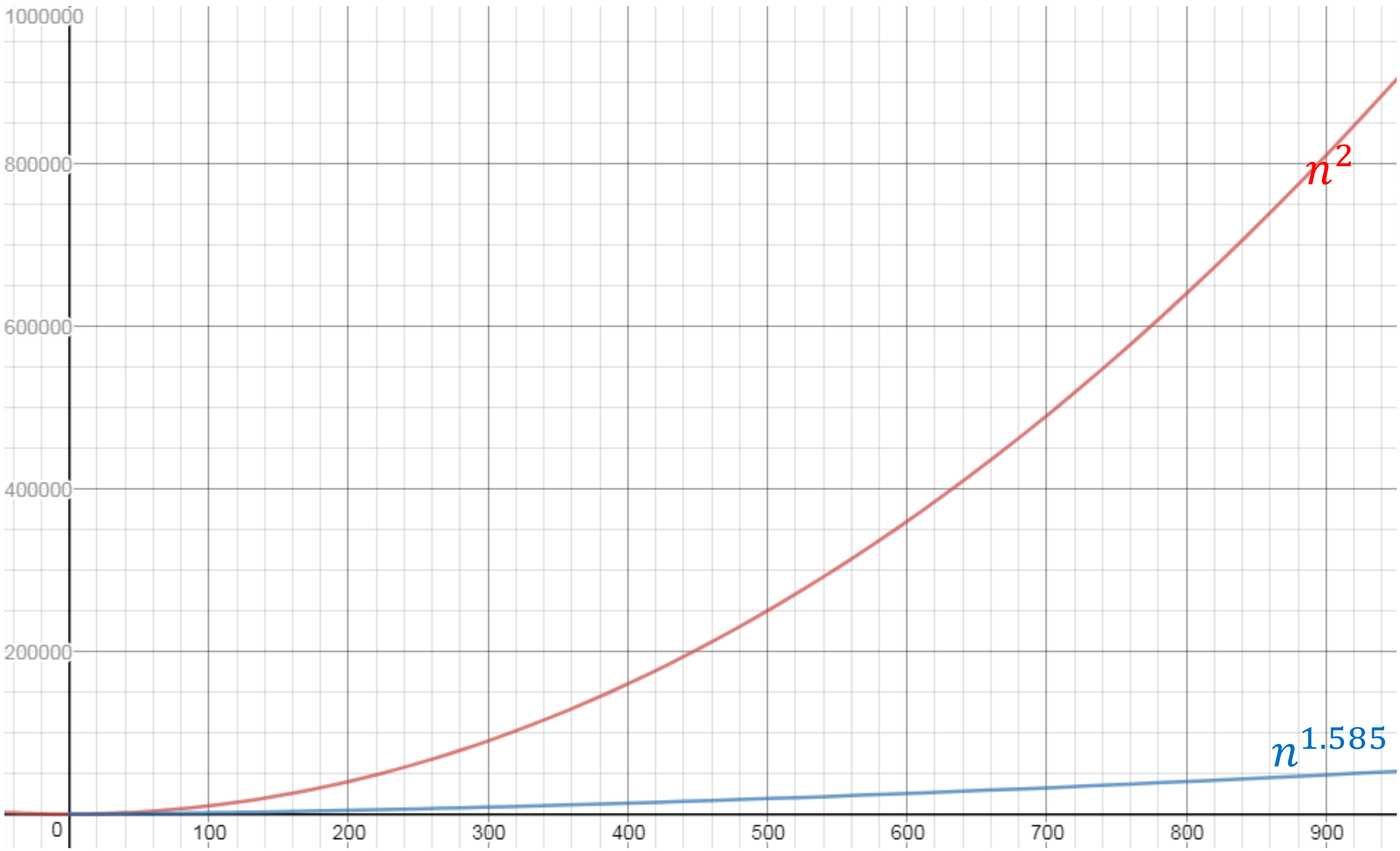
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

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

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

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

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


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

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

Technically, called *strong induction*

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

Inductive step: show $P(x_0) \Rightarrow P(x_0 + 1)$

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 <u>base case</u> or to the <u>inductive hypothesis</u>

Karatsuba Guess and Check (Loose)

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

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

Base cases: $T(1) = 8 \leq 3000$

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

… up to some small k

Hypothesis: $\forall n \leq x_0, T(n) \leq 3000n^{1.6}$

Inductive step: Show that $T(x_0 + 1) \leq 3000(x_0 + 1)^{1.6}$

Karatsuba Guess and Check (Loose)

Karatsuba Guess and Check (Loose)