CS3000: Algorithms & Data
Jonathan Ullman

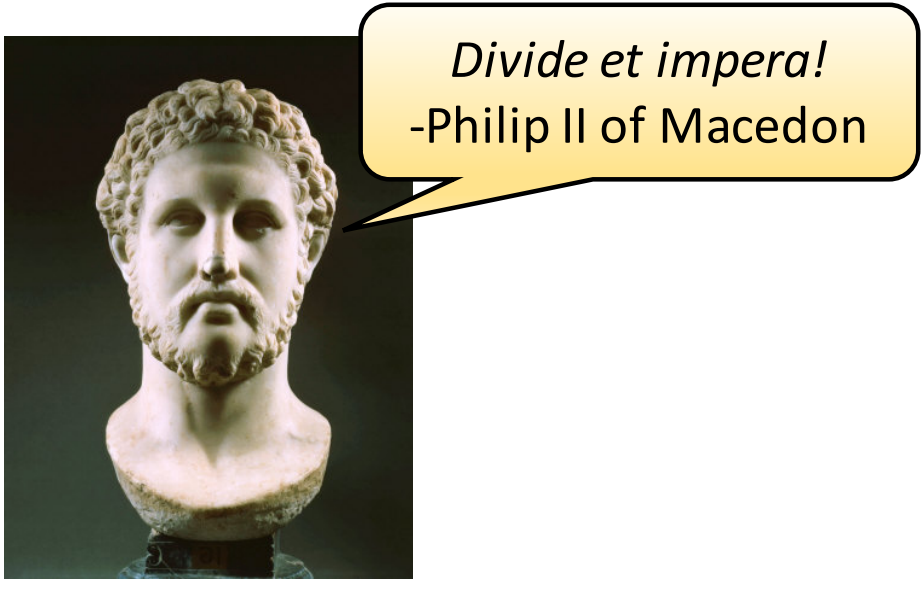
Lecture 4:

- Divide and Conquer: Karatsuba
- Solving Recurrences

Sep 18, 2018

Mergesort Wrapup

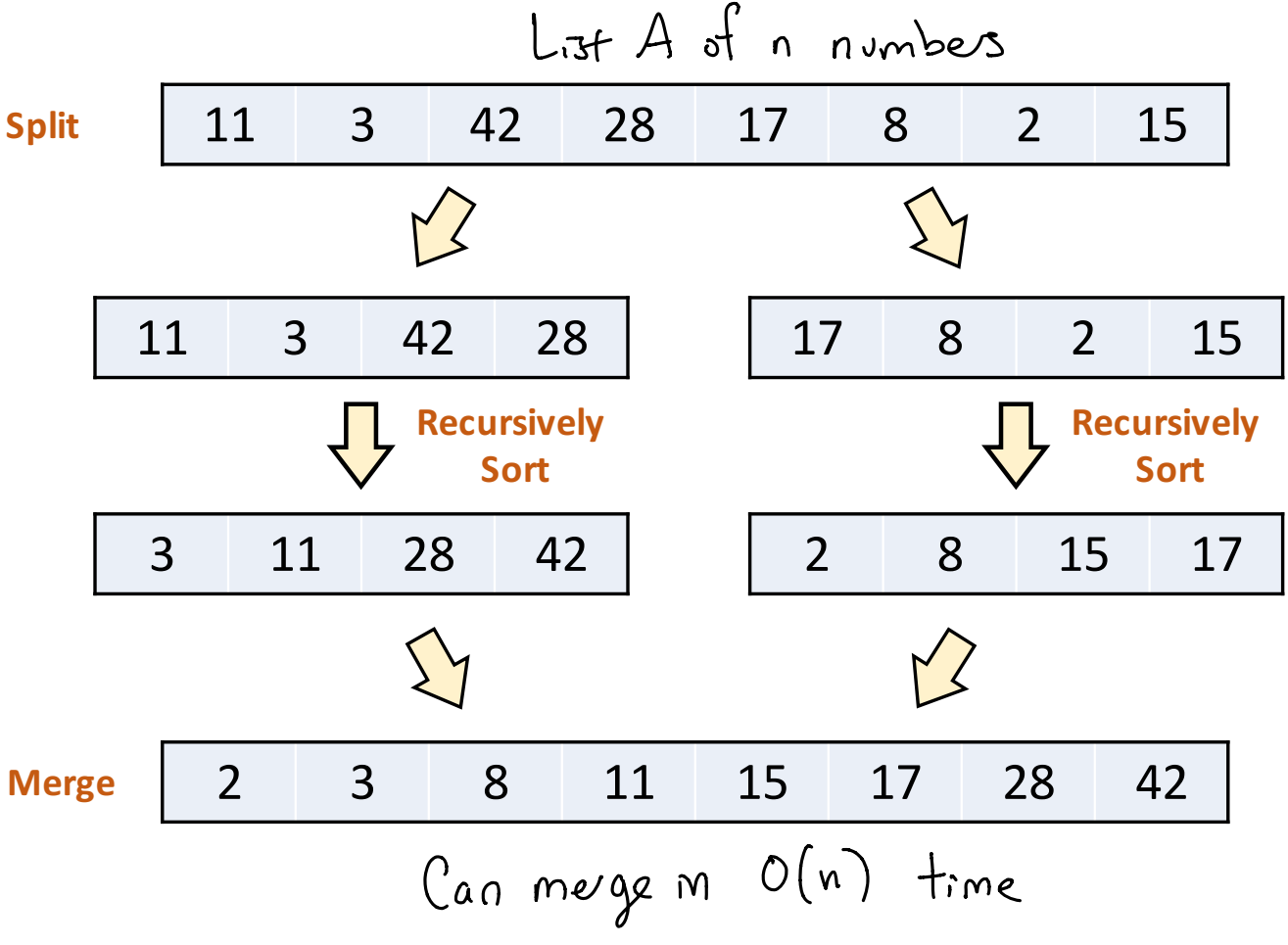
Divide and Conquer Algorithms



- Split your problem into smaller subproblems
- Recursively solve each subproblem
- Combine the solutions to the subprobelms

Useful when combining solutions is easier than solving from scratch

Mergesort


List A of n numbers
Split
11 3 42 28 17 8 2 15
11 3 42 28
17 8 2 15
Recursively Sort
Recursively Sort
3 11 28 42
2 8 15 17
Merge
2 3 8 11 15 17 28 42
Can merge in O(n) time

Running Time of Mergesort

$T(n)$: time to sort lists of size n

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

$T(1) = C$

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

```
MergeSort(A):
  If (n = 1): Return A

  Let   m ← ⌈n/2⌉
        L ← A[1:m]
        R ← A[m+1:n]

  Let L ← MergeSort(L)
  Let R ← MergeSort(R)
  Let A ← Merge(L,R)

  Return A
```

O(1)

O(n)

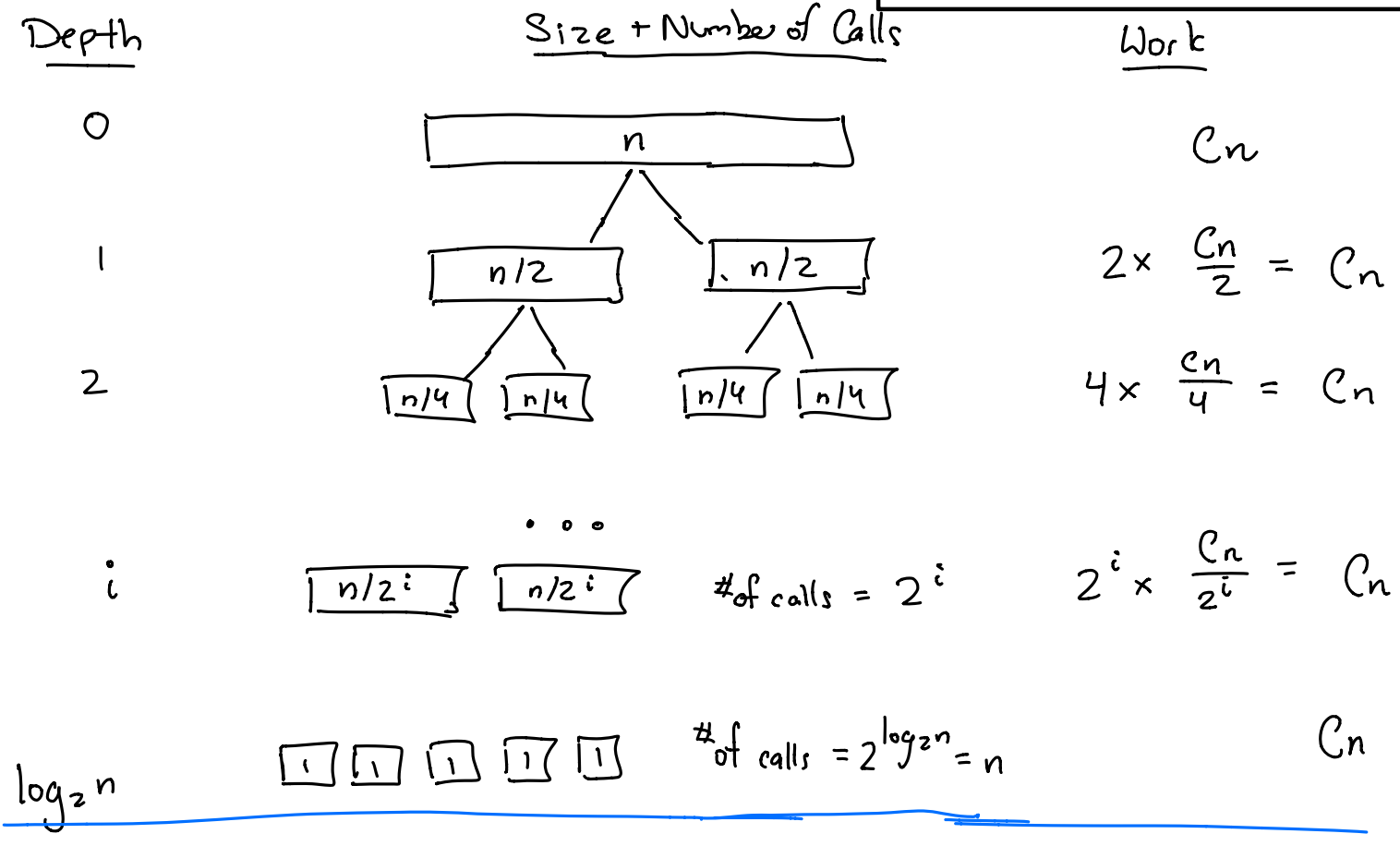
$2 \times T\left(\frac{n}{2}\right)$

O(n)

O(1)

Recursion Tree

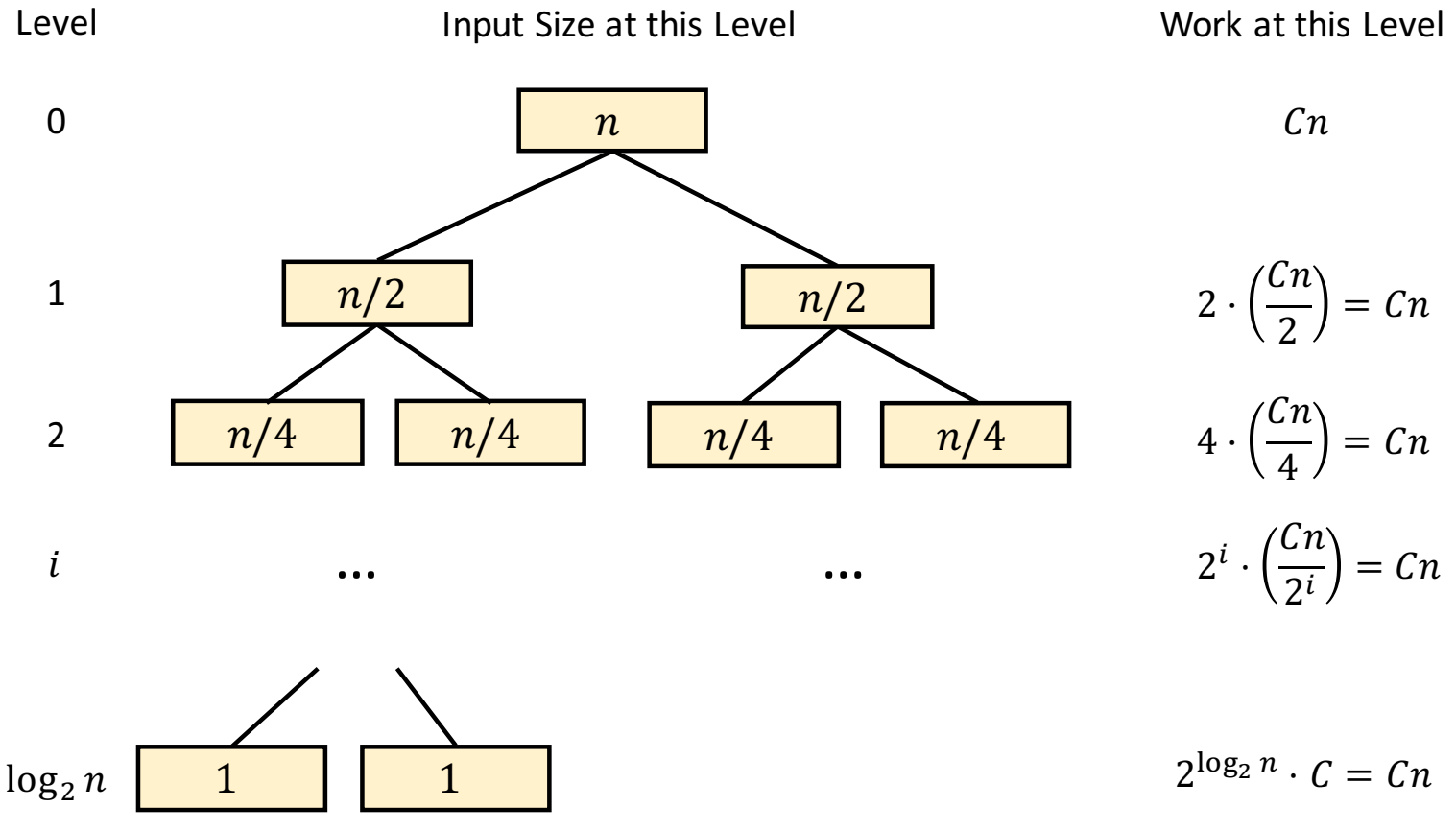
$$T(n) = 2 \cdot T(n/2) + Cn$$
$$T(1) = C$$



total work is $T(n) = Cn(\log_2 n + 1) = \Theta(n \log n)$

Recursion Tree

$$T(n) = 2 \cdot T(n/2) + Cn$$
$$T(1) = C$$



Proof by Induction

$$T(n) = 2 \cdot T(n/2) + Cn$$
$$T(1) = C$$

- **Claim:** $T(n) = Cn \log_2 2n$

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

$$= 2\left(C \cdot \frac{n}{2} \cdot \log_2 n\right) + Cn$$

$$= Cn \log_2 n + Cn$$

$$= Cn\left(\log_2 n + 1\right)$$

$$= Cn \log_2 2n$$

Mergesort Summary

- Sort a list of n numbers in $O(n \log n)$ time
  - Can actually sort anything that allows comparisons
  - Any comparison based algorithm is $\Omega(n \log n)$ time
- Divide-and-conquer approach
  - Break the list into two halves, sort each one and merge
  - Key Fact: merging is easier than sorting
- Proof of correctness
  - Proof by induction
- Analysis of running time
  - Solve a recurrence

Integer Multiplication: Karatsuba's Algorithm

Addition

- Given n-digit numbers x, y output $x + y$

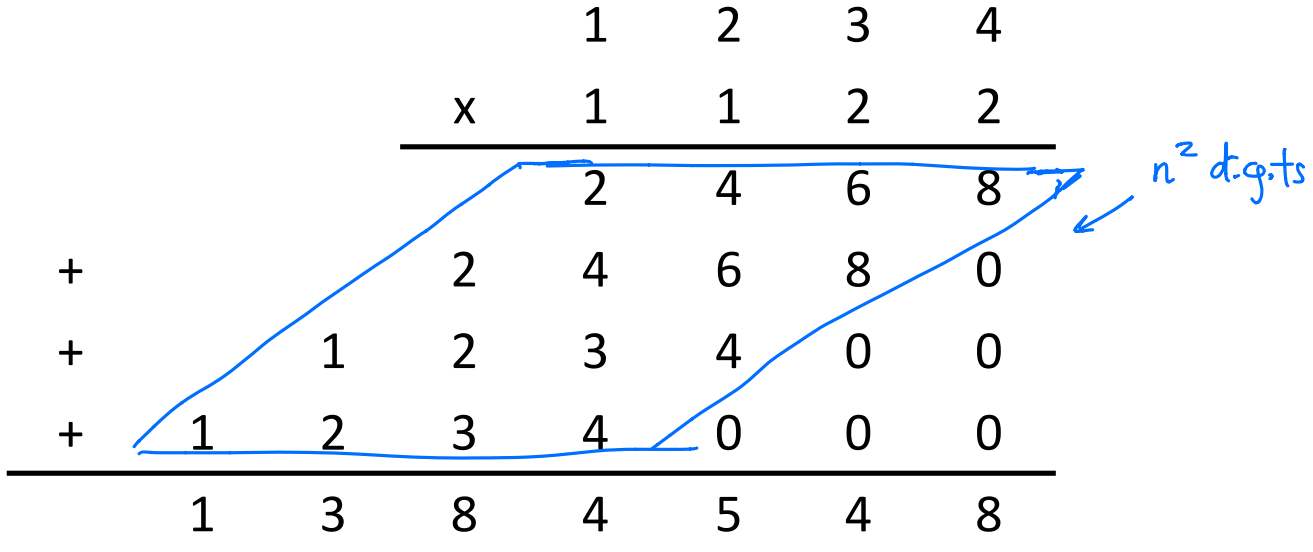
	1	2	3	4
+	1	1	2	2
=	2	3	5	6

Basic Operation: adding digits w/ carry

Running Time: $O(n)$

Multiplication

- Given n-digit numbers x, y output $x \cdot y$



Running Time: $\Theta(n^2)$ time

Divide and Conquer Multiplication

	1	2	3	4
x	1	1	2	2

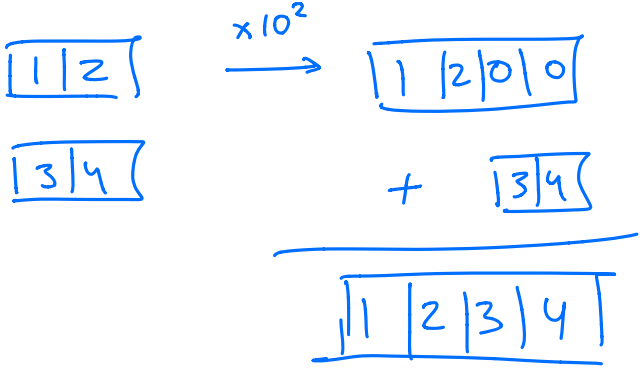
$$x = 10^2 \cdot 12 + 34$$

$$y = 10^2 \cdot 11 + 22$$

	a	b
x	c	d

$$x = 10^{n/2}a + b$$

$$y = 10^{n/2}c + d$$



Observation:

Multiplying by 10^n is "free"

Divide and Conquer Multiplication

	1	2	3	4
x	1	1	2	2

$$x = 10^2 \cdot 12 + 34$$
$$y = 10^2 \cdot 11 + 22$$

	a	b
x	c	d

$$x = 10^{n/2}a + b$$
$$y = 10^{n/2}c + d$$

$$x \cdot y = (10^2 \cdot 12 + 34)(10^2 \cdot 11 + 22)$$

$$= 10^4(12 \times 11) + 10^2(12 \times 22 + 11 \times 34) + 34 \times 22$$

Multiply two 4-dig numbers → Four mults of 2-digit numbers + Shifts + Adds

Divide and Conquer Multiplication

	$\boldsymbol{a}$	$\boldsymbol{b}$
x	$\boldsymbol{c}$	$\boldsymbol{d}$

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$
$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

$$\boldsymbol{x} \cdot \boldsymbol{y} = (10^{n/2}\boldsymbol{a} + \boldsymbol{b})(10^{n/2}\boldsymbol{c} + \boldsymbol{d})$$
$$= 10^{n}\boldsymbol{ac} + 10^{n/2}(\boldsymbol{ad} + \boldsymbol{bc}) + \boldsymbol{bd}$$

- Four $n/2$-digit mults, three n-digit adds
  - Multiplying by 10^n is "free" because it's a shift
- Recurrence: $T(n) = 4T\left(\frac{n}{2}\right) + 3n$

Divide and Conquer Multiplication

$$T(n) = 4 \cdot T(n/2) + 3n$$
$$T(1) = 1$$

- **Claim:** $T(n) \geq n^2$

$$T(n) = 4 \times T\left(\frac{n}{2}\right) + 3n$$
$$\geq 4 \times \left(\frac{n}{2}\right)^2 + 3n$$
$$= n^2 + 3n \geq n^2$$

Karatsuba's Algorithm

	a	b
x	c	d

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$

$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

→ Do not need ad and bc

$$x \cdot y = 10^{n}\boldsymbol{ac} + 10^{n/2}(\boldsymbol{ad} + \boldsymbol{bc}) + \boldsymbol{bd}$$

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

- Only three $n/2$-digit mults (plus some adds)!

Karatsuba's Algorithm

```
Karatsuba(x,y,n):
  If (n = 1): Return x·y                    // Base Case

  Let m ← ⌈n/2⌉                             // Split
  Write x = 10^m a + b,  y = 10^m c + d

  Let e ← Karatsuba(a,c,m)                  // Recurse
      f ← Karatsuba(b,d,m)
      g ← Karatsuba(b-a,c-d,m)

  Return 10^2m e + 10^m (e + f + g) + f     // Merge
```

Correctness of Karatsuba

- **Claim:** The algorithm **`Karatsuba`** is correct
- $\forall n \quad \forall x, y$ with n-digits $\quad Karatsuba(x, y, n) = x \cdot y$

Proof:

Inductive Hypotheses: $H(n) = \forall x, y$ with n digits, $K(x, y, n) = x \cdot y$

Base Case: $H(1)$ is true, obviously

Correctness of Karatsuba

- **Claim:** The algorithm **`Karatsuba`** is correct

Proof Cont'd:

Inductive Step: Assume $H(1) \wedge \ldots \wedge H(n)$, meaning that Karatsuba is correct for all x, y with $\leq n$ digits. Want to show $H(n+1)$. Consider any x, y with $n+1$ digits.

- By definition, $a, b, c, d, b-a, c-d$ all have $\leq n$ digits.
- Therefore, by $H(1) \wedge \ldots \wedge H(n)$, $e = a \cdot b$, $f = c \cdot d$, $g = (b-a)(c-d)$.
- Therefore, $K(x, y, n+1) = 10^{2m} e + 10^{m} (e+f+g) + f$

$= 10^{2m} a \cdot b + 10^{m} (ad + bc) + c \cdot d = x \cdot y$ □

Running Time of Karatsuba

$T(n)$: time to mult n-digit numbers

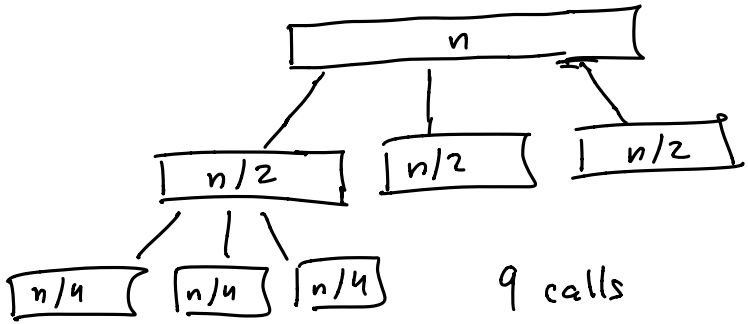
$T(n) = 3T(\frac{n}{2}) + Cn$

```
Karatsuba(x,y,n):
  If (n = 1): Return x · y                    O(1)

  Let m ← ⌈n/2⌉                               O(n)
  Write x = 10^m a + b,  y = 10^m c + d

  Let e ← Karatsuba(a,c,m)                    3×T(n/2)
      f ← Karatsuba(b,d,m)
      g ← Karatsuba(b−a,c−d,m)

  Return 10^{2m} e + 10^m (e + f + g) + f
```

6 additions

Recursion Tree

$$T(n) = 3 \cdot T(n/2) + Cn$$
$$T(1) = C$$



$$T(n) = Cn \times \sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i \approx Cn \times \left(\frac{3}{2}\right)^{\log_2 n} = C3^{\log_2 n} = Cn^{\log_2 3}$$

Geometric Series

$r = 1 \Rightarrow S = (\ell+1) \cdot r$

$r \neq 1$

- Series $S = \sum_{i=0}^{\ell} r^i$

$$S = 1 + r + r^2 + \cdots + r^{\ell}$$
$$rS = \quad r + r^2 + \cdots + r^{\ell} + r^{\ell+1}$$

$r > 1 \Rightarrow S \approx r^{\ell}$

$r < 1 \Rightarrow S \to \frac{1}{1-r}$

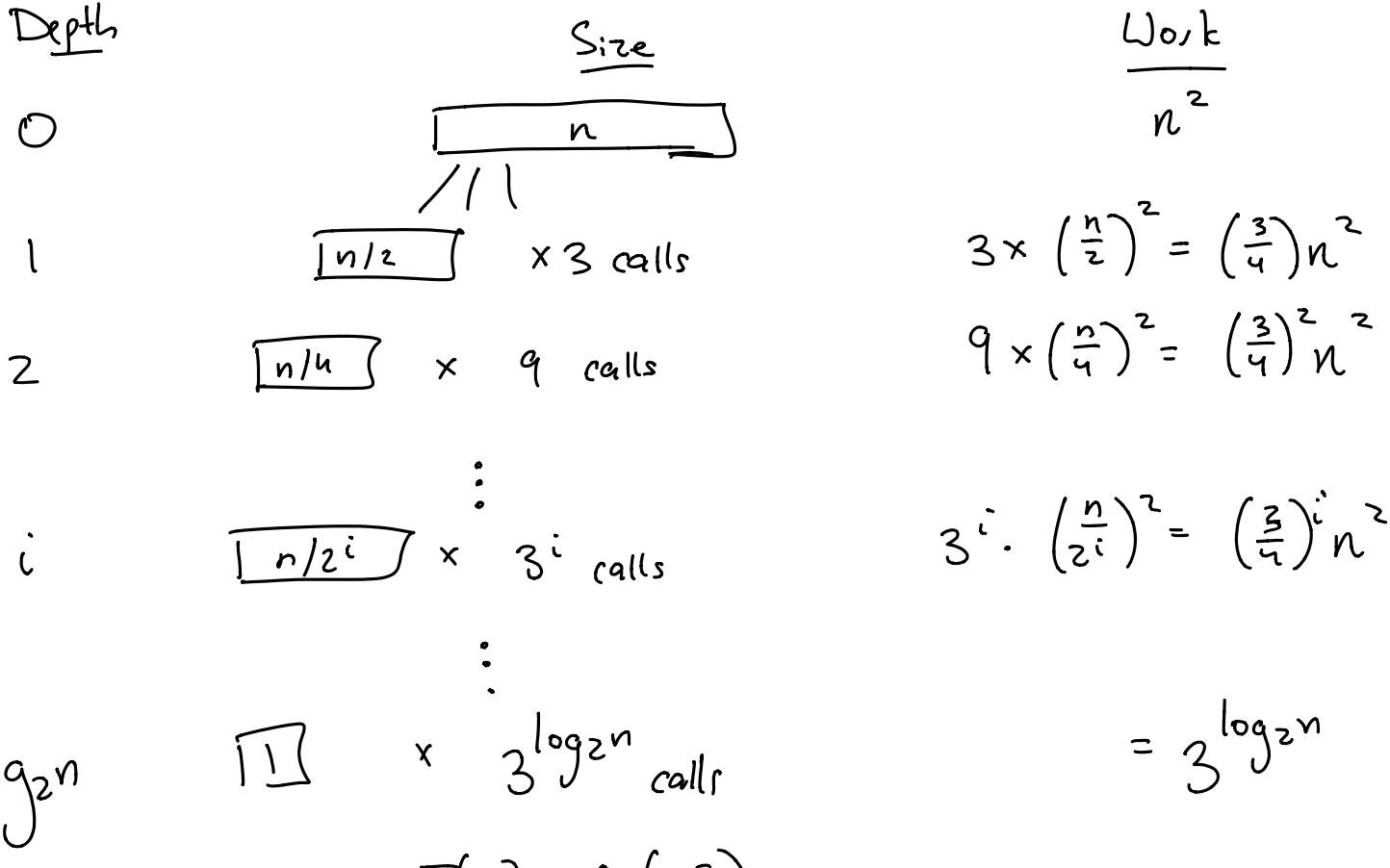
- Solution $S = \frac{1-r^{\ell+1}}{1-r} = \frac{r^{\ell+1}-1}{r-1}$

Karatsuba Wrapup

- Multiply n digit numbers in $O(n^{1.59})$ time
  - Improves over naïve $O(n^2)$ time algorithm
  - **Fast Fourier Transform:** multiply in $\approx O(n \log n)$ time
- Divide-and-conquer approach
  - Uses a clever algebraic trick to split
  - **Key Fact:** adding is faster than multiplying
- Prove correctness via induction
- Analyze running time via recursion tree
  - $T(n) = 3T(n/2) + Cn$

Ask the Audience!

- $T(n) = 3T(n/2) + n^2$
- $T(1) = 1$



$$\sum_{i=0}^{\log_2 n} \left(\frac{3}{4}\right)^i n^2 = n^2 \sum_{i=0}^{\log_2 n} \left(\frac{3}{4}\right)^i$$

$$\leq n^2 \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i$$

$$= n^2 \cdot \frac{1}{1-\frac{3}{4}} = 4n^2$$

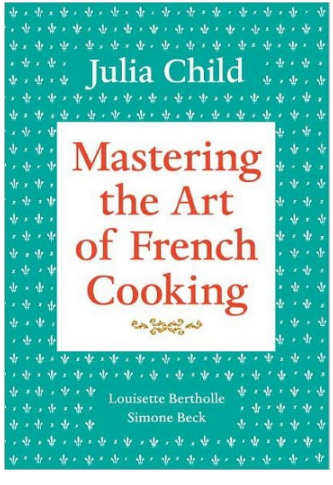
Solving Recurrences:
“The Master Theorem”

The "Master Theorem"

Mergesort: $a=2$, $b=2$, $d=1$

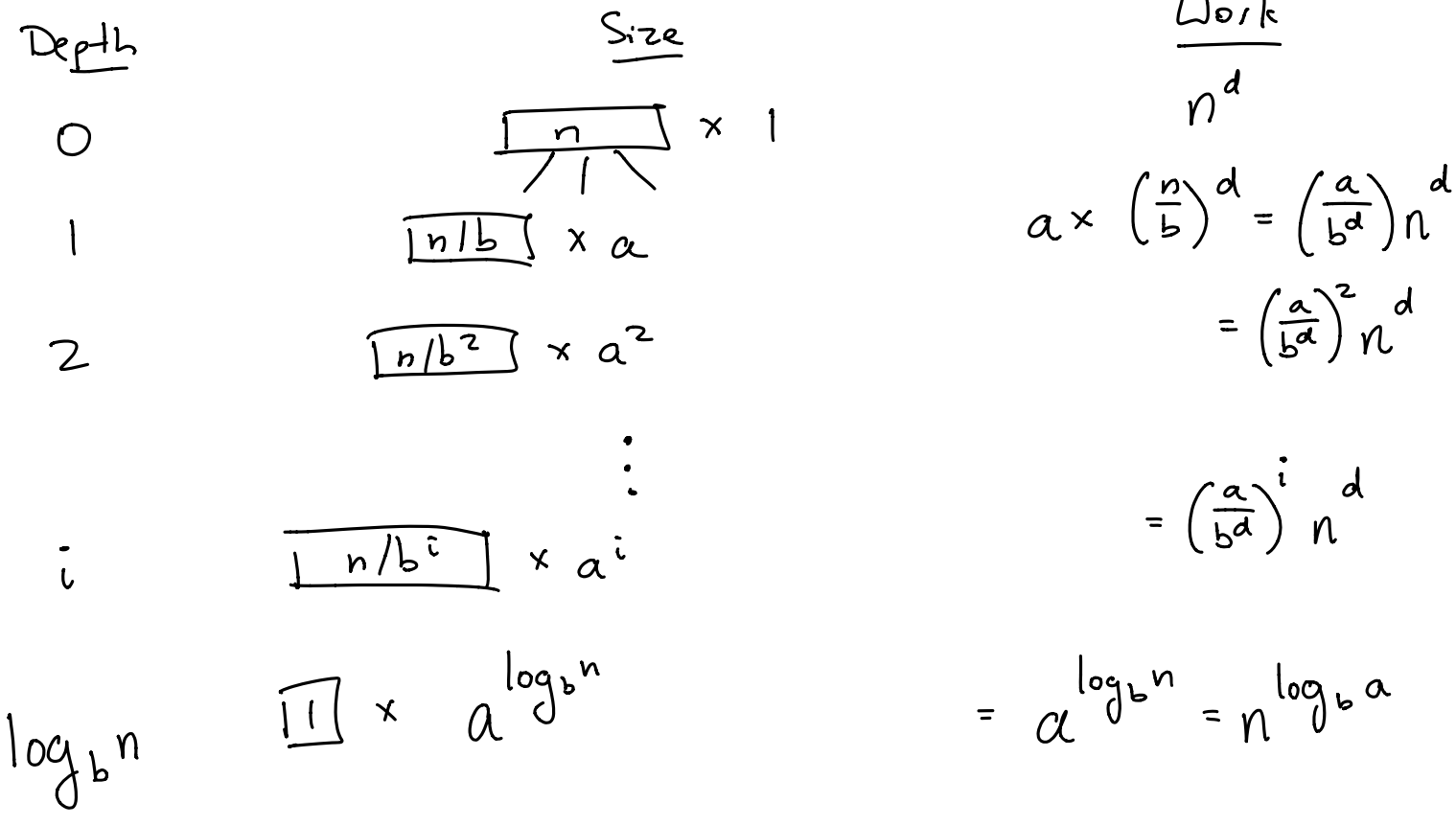
Karatsuba: $a=3$, $b=2$, $d=1$

- Generic divide-and-conquer algorithm:
  - Split into $\boldsymbol{a}$ pieces of size $\frac{\boldsymbol{n}}{\boldsymbol{b}}$ and merge in time $O(n^{\boldsymbol{d}})$
- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$



Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$
- $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1$



Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$
- $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) = 1$

$$S = \sum_{i=0}^{\log_b n} n^d \left(\frac{a}{b^d}\right)^i = n^d \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i$$

Case 1: $\left(\frac{a}{b^d}\right) > 1$ $\quad S = O\left(n^d \cdot \left(\frac{a}{b^d}\right)^{\log_b n}\right)$

$= O(n^{\log_b a})$

Case 2: $\left(\frac{a}{b^d}\right) = 1$ $\quad S = n^d \cdot \log_b n = O(n^d \log n)$

Case 3: $\left(\frac{a}{b^d}\right) < 1$ $\quad S \leq n^d \cdot \frac{1}{1 - \frac{a}{b^d}} = O(n^d)$

Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$
- $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1$

The "Master Theorem"

independent of n

- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1 : \; T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) = 1 : \; T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1 : \; T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$

Ask the Audience!

- Use the Master Theorem to Solve:
  - $T(n) = 16 \cdot T\left(\frac{n}{4}\right) + n^2$ $a=16$, $b=4$, $d=2$; $\frac{16}{4^2}=1 \Rightarrow T(n)=\Theta(n^2 \log n)$
  - $T(n) = 21 \cdot T\left(\frac{n}{5}\right) + n^2$ $a=21$, $b=5$, $d=2$; $\frac{21}{5^2}<1 \Rightarrow T(n)=\Theta(n^2)$
  - $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 1$ $a=2$, $b=2$, $d=0$; $\frac{2}{2^0}>1 \Rightarrow T(n)=\Theta(n^{\log_2 2})=\Theta(n)$
  - $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + 1$ $a=1$, $b=2$, $d=0$; $\frac{1}{2^0}=1 \Rightarrow T(n)=\Theta(n^0 \log n)=\Theta(\log n)$

The "Master Theorem"

- **Even More General:** all recurrences of the form
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + f(n)$
- Three cases:
  - $f(n) = O(n^{(\log_{\boldsymbol{b}} \boldsymbol{a})-\varepsilon})$:
    - $T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$
  - $f(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$:
    - $T(n) = \Theta(f(n) \cdot \log n)$
  - $f(n) = \Omega(n^{(\log_{\boldsymbol{b}} a)+\varepsilon})$ **AND** $\boldsymbol{a} f\left(\frac{n}{\boldsymbol{b}}\right) \leq C f(n)$ for $C < 1$
    - $T(n) = \Theta\left(f(n)\right)$