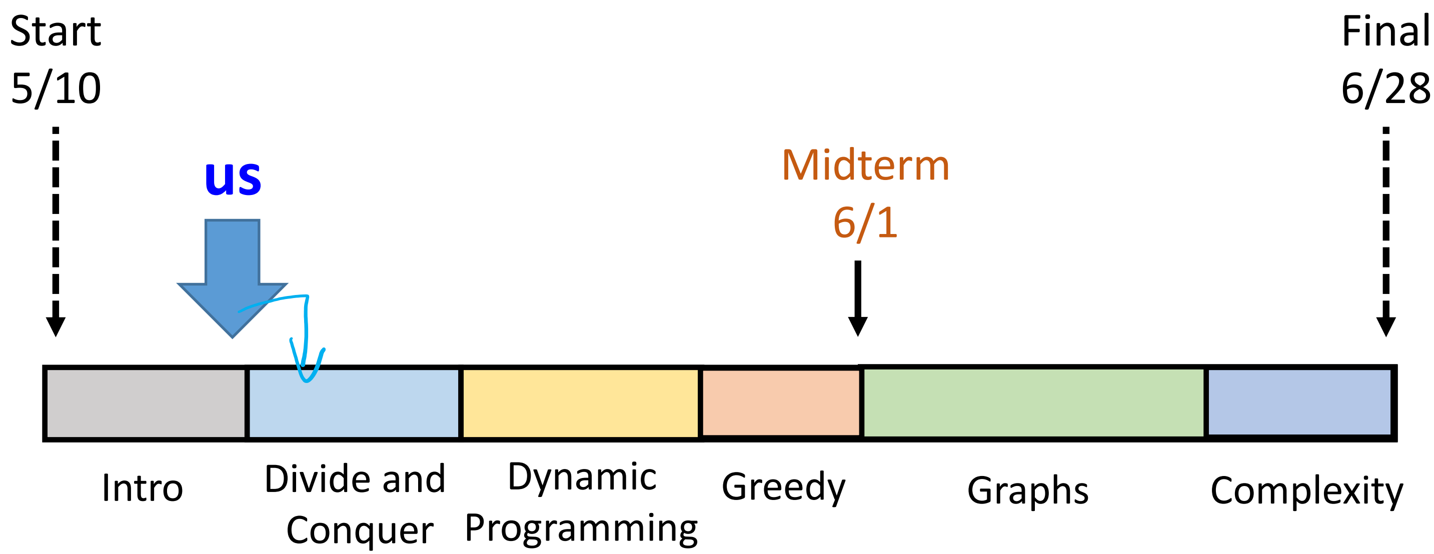
(record)
Q1
Rec 2

CS3000: Algorithms & Data
Drew van der Poel

Lecture 5
- Divide & Conquer: Merge Sort
- Divide & Conquer: Karatsuba's

May 17, 2021

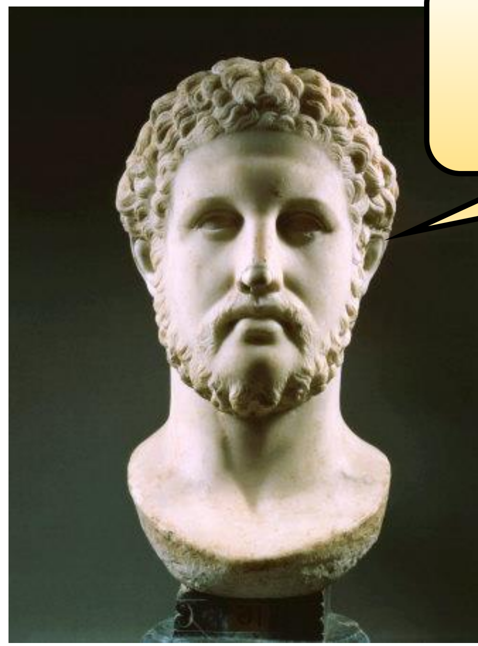
Outline



Last class: stable matching, asymptotic analysis, selection sort

Next class: divide and conquer: Karatsuba's + Master Thm.

Divide and Conquer Algorithms



Divide et impera!
-Philip II of Macedon

D+C Recipe

same as the original problem

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

Keep splitting

A Simple Algorithm: Selection Sort

Find the min. & swap with end

11	3	42	28	17	8	2	15

Repeat n-1 times

2	3	8	11	15	17	28	42

(asymptotic time complexity)

Running Time (# of operations):

Steps:

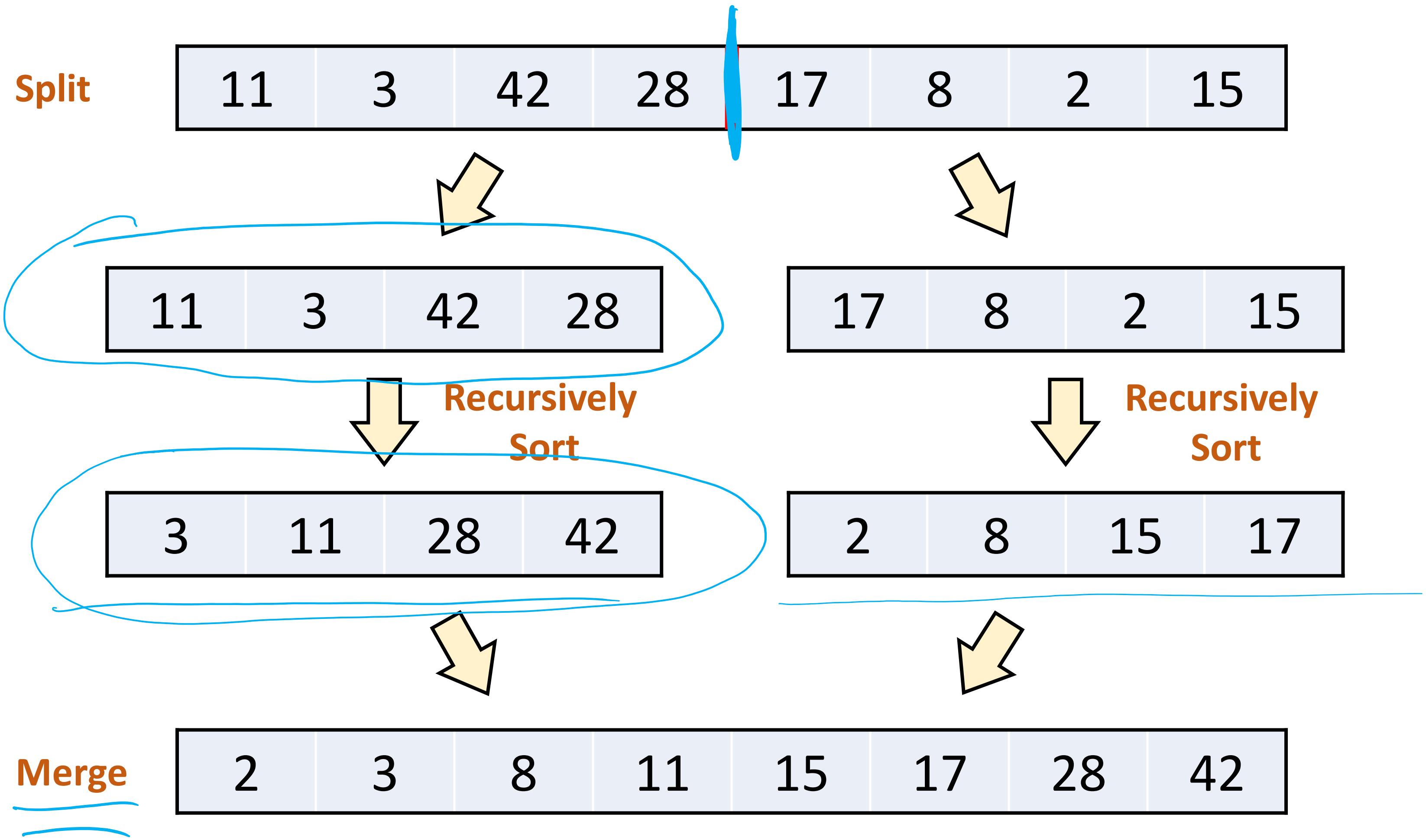
	# of ops.
scan	n
swap	1

total # of ops. n iterations

$n + 1 + n-1 + 1 + n-2 + 1 + \dots 1 + 1$

$$\sum_{i=1}^{n} i + n = \frac{n^2}{2} + \frac{n}{2} + n = O(n^2)$$

Divide and Conquer: Mergesort



Divide and Conquer: Mergesort

- **Key Idea:** If $\boldsymbol{L}, \boldsymbol{R}$ are sorted lists of length $n/2$, then we can merge them into a sorted list $\boldsymbol{A}$ of length n in time $\Theta(\ n\ )$
  - Merging two sorted lists is faster than sorting from scratch

n comparisons

n loops

n insertions

3	11	28	42	$\boldsymbol{L}$
2	8	15	17	$\boldsymbol{R}$

2	3	8	11	15	17	28	42	$\boldsymbol{A}$

Merging Pseudocode

```
Merge(L,R):
  Let n ← len(L) + len(R)
  Let A be an array of length n
  j ← 1, k ← 1,

  For i = 1,…,n:
    If (j > len(L)):                    // L is empty
      A[i] ← R[k], k ← k+1
    ElseIf (k > len(R)):                // R is empty
      A[i] ← L[j], j ← j+1
    ElseIf (L[j] <= R[k]):              // L is smallest
      A[i] ← L[j], j ← j+1
    Else:                               // R is smallest
      A[i] ← R[k], k ← k+1

  Return A
```

$O(1)$ 1 1 2

// j is the current index for L

k " " R

n loops

$3 \le$ # of ops ≤ 5

1

TOTAL # of ops: $4 + Cn + 1 = \Theta(n)$

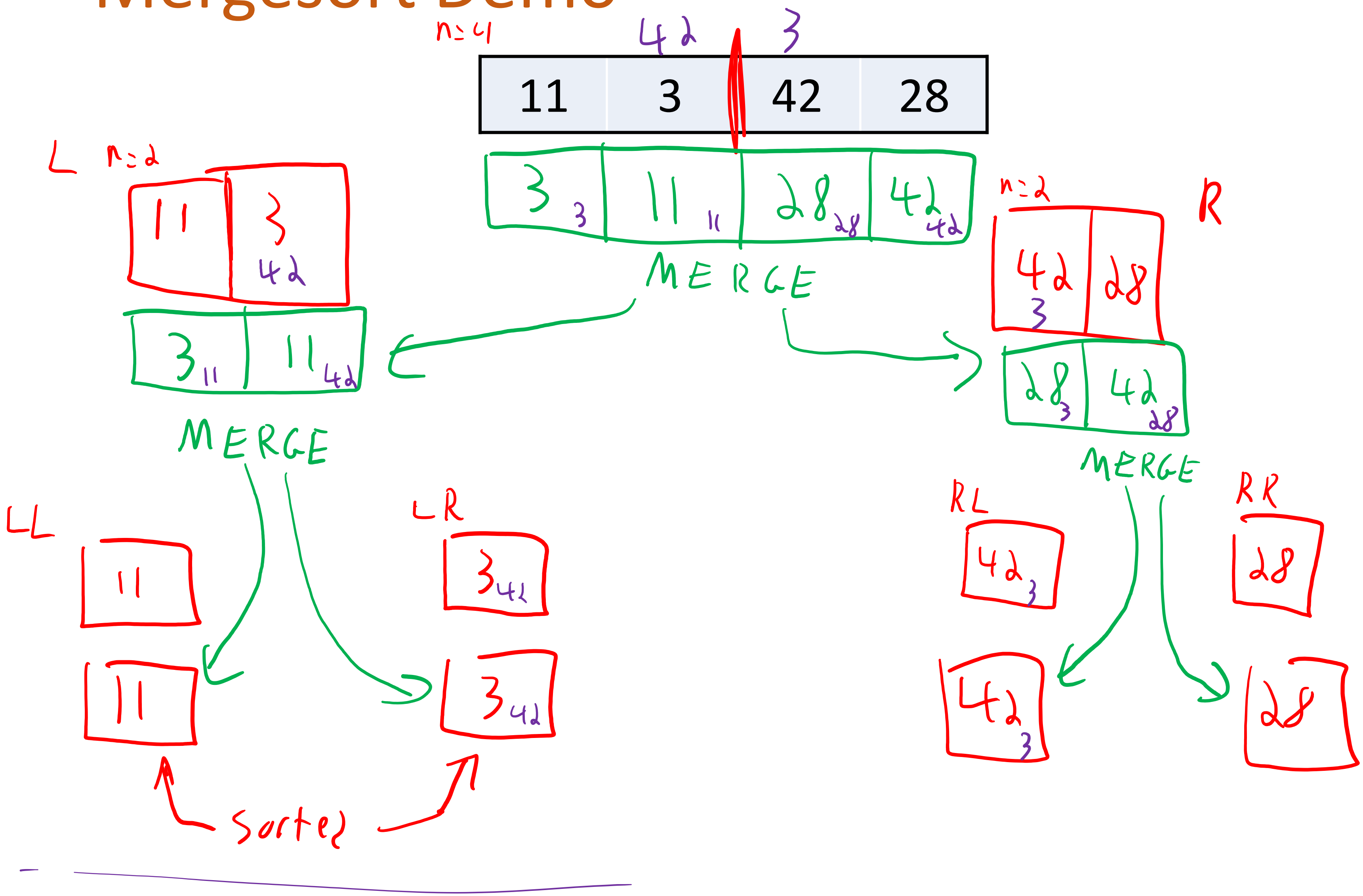
MergeSort

```
MergeSort(A):
   If (len(A) = 1): Return A     // Base Case

   Let m ← ⌈len(A)/2⌉            // Split
   Let L ← A[1:m], R ← A[m+1:n]

   Let L ← MergeSort(L)          // Recurse
   Let R ← MergeSort(R)

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

   Return A
```

sorted

sorted

Mergesort Demo



Correctness of Mergesort

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

- H(n): Mergesort correctly sorts all lists of size $n \in \mathbb{Z}^+$

- Base Case:

  H(1) – trivial

Correctness of Mergesort

- Inductive step:

$H(1) \wedge \ldots \wedge H(k-1) \to H(k)$

L and R have length $\leq \lceil \frac{k}{2} \rceil$

(m and m(-1), respectively)

$\therefore$ L and R are

correctly sorted by Mergesort (by IH) $\lceil \frac{k}{2} \rceil < k$ $(k \geq 2)$

Because merge correctly merges 2 sorted lists into 1 sorted list $\longrightarrow$ A is sorted ✓

* You can assume Merge is correct *

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

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

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

  Return A
```

Running Time of Mergesort

T(1) = $1 = c = O(1)$

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

size of list

2 calls

half the size

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

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

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

  Return A
```

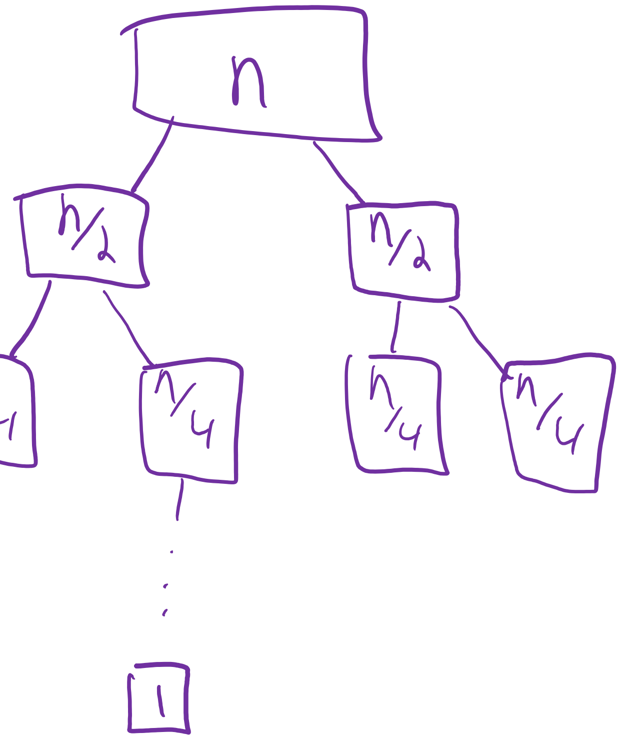
$\Theta(n)$

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

Recursion Trees

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

level	piece size	# of pieces	work @ level
0	$n = n/2^0$	1	cn
1	$n/2 = \frac{n}{2^1}$	2	$c\frac{n}{2} + c\frac{n}{2} = cn$
2	$n/4 = \frac{n}{2^2}$	4	$4\left(\frac{cn}{4}\right) = cn$
i	$n/2^i$	2^i	cn



$$\underline{\underline{\log_2 n}}$$

$$\frac{n}{2^i} = 1 \longrightarrow n = 2^i$$
$$\log_2 n = i$$

$$\frac{cn}{cn}$$

Running Time of Mergesort

Total work: $\sum_{i=0}^{last\ level} work\ at\ level\ i$

last level: $\log_2 n$

work @ level i: cn

$$\sum_{i=0}^{\log_2 n} cn = (\log_2 n + 1)\, cn$$

$$= cn \log_2 n + cn = O(n \log_2 n)$$

- Problems: counting students, stable matching, sorting

- Alg. techniques: divide & conquer

- Analysis: asymptotic analysis, **recursion trees**

- Proof techniques: (strong) induction, contradiction

Recursion Trees

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

Level	Input Size at this Level	Work at this Level
0	n	Cn
1	$n/2$ $\quad$ $n/2$	$2 \cdot \left(\frac{Cn}{2}\right) = Cn$
2	$n/4$ $\quad$ $n/4$ $\quad$ $n/4$ $\quad$ $n/4$	$4 \cdot \left(\frac{Cn}{4}\right) = Cn$
i	… $\quad$ …	$2^i \cdot \left(\frac{Cn}{2^i}\right) = Cn$
$\log_2 n$	1 $\quad$ 1	$2^{\log_2 n} \cdot C = Cn$

Mergesort Summary

- Sort a list of n numbers in $\Theta(n \log n)$ time
  - Can actually sort anything that allows comparisons
  - No comparison based algorithm can be (much) faster
- Divide-and-conquer
  - 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
  - Recurrences & recursion trees

Integer Multiplication: Karatsuba’s Algorithm

Addition

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

	1			
	1	~~2~~ 9	3	4
+	1	1	2	2
=	~~2~~	~~3~~ 0	5	6
	3			

$2n - 2 + 1 =$

$2n - 1 =$

Running Time:

$n \leq \#\text{ of additions} \leq 2(n-1) + 1$

$0 \leq \#\text{ of carries} \leq n - 1$

Total: $\Theta(n)$

Multiplication

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

$$
\begin{array}{cccccccc}
 & & & & 1 & 2 & 3 & \cancel{4}\,8 \\
 & & & \times & 1 & 1 & 2 & 2 \\
\hline
 & 0 & 0 & 0 & 2 & 4 & 6 & 8 \\
+ & 0 & 0 & 2 & 4 & 6 & 8 & 0 \\
+ & 0 & 1 & 2 & 3 & 4 & 0 & 0 \\
+ & 1 & 2 & 3 & 4 & 0 & 0 & 0 \\
\hline
 & 1 & 3 & 8 & 4 & 5 & 4 & 8
\end{array}
$$

n^2 mults

$\leq n-1$ adds.

$n-1$ adds. or $\Theta(n)$-digit #'s

$n-1$ additions of $\Theta(n)$-digit #s

$cn \cdot (n-1) = cn^2 - cn$

Running Time: $O(n^2)$

$cn^2 + cn^2 - cn =$

- Problems: counting students, stable matching, sorting, **n-digit mulitiplication**
- Alg. techniques: divide & conquer
- Analysis: asymptotic analysis, recursion trees
- Proof techniques: (strong) induction, contradiction

Divide and Conquer Multiplication

	1	2	3	4
x	1	1	2	2

$1234 =$

$1122 =$

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

ab =

cd =

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 & some shifts
- Recurrence: $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$

Divide and Conquer Multiplication

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

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

Karatsuba's Algorithm

	a	b
x	c	d

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

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

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

- Key Identity
  - $(\boldsymbol{b} - \boldsymbol{a})(\boldsymbol{c} - \boldsymbol{d}) =$

- Only three $n/2$-digit mults (plus some adds & shifts)!
  - 1.
  - 2.
  - 3.

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

H(n):

Base:

Correctness of Karatsuba

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

Inductive: H(1) & H(2) & … & H(k-1) -> H(k)

Running Time of Karatsuba

`Karatsuba(x,y,n):`

`If (n = 1): Return` $x \cdot y$

`Let` $m \leftarrow \lceil n/2 \rceil$

`Write` $x = 10^m a + b$, $y = 10^m c + d$

`Let e ← Karatsuba(a,c,m)`

`f ← Karatsuba(b,d,m)`

`g ← Karatsuba(b-a,c-d,m)`

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

Recursion Tree

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

Geometric Series

- Series (r≠1, r>0) $S = \sum_{i=0}^{\ell} r^i$

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

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

$$S\,(1 \; - r) = S \; - rS = 1 \; - r^{\ell+1}$$

$$S\,(r \; - 1) = rS \; - S = r^{\ell+1} - 1$$

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

- $S = \Theta(1)$ when $r < 1$
  $S = \Theta(r^\ell)$ when $r > 1$

Karatsuba Wrapup

- Multiply n digit numbers in $O\left(n^{1.59}\right)$ time
  - Improves over naïve $O\left(n^2\right)$ 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
  - $T(n) = 3T(n/2) + Cn$
- We will generally assume our inputs have *O(1)* digits