Divide & conquer: Kartsuba's Algorithm and Linear Time Selection

Lecture 11

Thursday, October 3, 2024

LaTeXed: October 8, 2024 20:53

11.1
Problem statement: Multiplying numbers + a slow algorithm

The Problem: Multiplying numbers

Given two large positive integer numbers b and c, with n digits, compute the number $b * c$.

Rhind Mathematical Papyrus

Roughly 3870 years ago

"Accurate reckoning for inquiring into things, and the knowledge of all things, mysteries ... all secrets"

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	$34 + 1$	76
76	34	
152	17	
152	$16 + 1$	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	**35**	
76	**34 + 1**	**76**
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	**35**	
76	**34 + 1**	**76**
76	**34**	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	34 + 1	76
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	**35**	
76	**34 + 1**	**76**
76	**34**	
152	**17**	
152	**16 + 1**	**152**
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	34 + 1	76
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	34 + 1	76
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	**35**	
76	**34 + 1**	**76**
76	**34**	
152	**17**	
152	**16 + 1**	**152**
152	**16**	
304	**8**	
608	**4**	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	34 + 1	76
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	34 + 1	76
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

Egyptian multiplication: 1850BC (3870 years ago?)

From the hieratic Moscow and Rhind Mathematical Papyri

76	35	
76	34 + 1	76
76	34	
152	17	
152	16 + 1	152
152	16	
304	8	
608	4	
1216	2	
2432	1	2432
		2660

The problem: Multiplying Numbers

Problem Given two n-digit numbers x and y, compute their product.

Grade School Multiplication

Compute "partial product" by multiplying each digit of y with x and adding the partial products.

$$
\begin{array}{r}
3141 \\
\times 2718 \\
\hline
25128 \\
3141 \\
21987 \\
6282 \\
\hline
8537238
\end{array}
$$

Time Analysis of Grade School Multiplication

1. Each partial product: $\boldsymbol{\Theta(n)}$
2. Number of partial products: $\boldsymbol{\Theta(n)}$
3. Addition of partial products: $\boldsymbol{\Theta(n^2)}$
4. Total time: $\boldsymbol{\Theta(n^2)}$

11.2
Multiplication using Divide and Conquer

Divide and Conquer

Assume n is a power of 2 for simplicity and numbers are in decimal.

Split each number into two numbers with equal number of digits

1. $b = b_{n-1}b_{n-2}\ldots b_0$ and $c = c_{n-1}c_{n-2}\ldots c_0$
2. $b = b_{n-1}\ldots b_{n/2}0\ldots 0 + b_{n/2-1}\ldots b_0$
3. $b(x) = b_L x + b_R$, where $x = 10^{n/2}$, $b_L = b_{n-1}\ldots b_{n/2}$ and $b_R = b_{n/2-1}\ldots b_0$
4. Similarly $c(x) = c_L x + c_R$ where $c_L = c_{n-1}\ldots c_{n/2}$ and $c_R = c_{n/2-1}\ldots c_0$

Example

$$\begin{aligned} 1234 \times 5678 &= (12x + 34) \times (56x + 78) \qquad \text{for} \quad x = 100. \\ &= 12 \cdot 56 \cdot x^2 + (12 \cdot 78 + 34 \cdot 56)x + 34 \cdot 78. \end{aligned}$$

$$\begin{aligned} 1234 \times 5678 &= (100 \times 12 + 34) \times (100 \times 56 + 78) \\ &= 10000 \times 12 \times 56 \\ &\quad +100 \times (12 \times 78 + 34 \times 56) \\ &\quad +34 \times 78 \end{aligned}$$

Divide and Conquer for multiplication

Assume n is a power of 2 for simplicity and numbers are in decimal.

1. $b = b_{n-1}b_{n-2}\dots b_0$ and $c = c_{n-1}c_{n-2}\dots c_0$
2. $b \equiv b(x) = b_L x + b_R$
   where $x = 10^{n/2}$, $b_L = b_{n-1}\dots b_{n/2}$ and $b_R = b_{n/2-1}\dots b_0$
3. $c \equiv c(x) = c_L x + c_R$ where $c_L = c_{n-1}\dots c_{n/2}$ and $c_R = c_{n/2-1}\dots c_0$

Therefore, for $x = 10^{n/2}$, we have

$$
\begin{aligned}
bc = b(x)c(x) &= (b_L x + b_R)(c_L x + c_R) \\
&= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\
&= 10^n b_L c_L + 10^{n/2}(b_L c_R + b_R c_L) + b_R c_R
\end{aligned}
$$

Divide and Conquer for multiplication

Assume n is a power of 2 for simplicity and numbers are in decimal.

1. $b = b_{n-1}b_{n-2}\dots b_0$ and $c = c_{n-1}c_{n-2}\dots c_0$
2. $b \equiv b(x) = b_L x + b_R$
   where $x = 10^{n/2}$, $b_L = b_{n-1}\dots b_{n/2}$ and $b_R = b_{n/2-1}\dots b_0$
3. $c \equiv c(x) = c_L x + c_R$ where $c_L = c_{n-1}\dots c_{n/2}$ and $c_R = c_{n/2-1}\dots c_0$

Therefore, for $x = 10^{n/2}$, we have

$$\begin{aligned} bc = b(x)c(x) &= (b_L x + b_R)(c_L x + c_R) \\ &= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\ &= 10^n b_L c_L + 10^{n/2}(b_L c_R + b_R c_L) + b_R c_R \end{aligned}$$

Time Analysis

$$bc = 10^n b_L c_L + 10^{n/2}(b_L c_R + b_R c_L) + b_R c_R$$

4 recursive multiplications of number of size $n/2$ each plus 4 additions and left shifts (adding enough 0's to the right)

$$T(n) = 4T(n/2) + O(n) \qquad T(1) = O(1)$$

$T(n) = \Theta(n^2)$. No better than grade school multiplication!

Time Analysis

$$bc = 10^n b_L c_L + 10^{n/2}(b_L c_R + b_R c_L) + b_R c_R$$

4 recursive multiplications of number of size $n/2$ each plus 4 additions and left shifts (adding enough 0's to the right)

$$T(n) = 4T(n/2) + O(n) \qquad T(1) = O(1)$$

$T(n) = \Theta(n^2)$. No better than grade school multiplication!

Time Analysis

$$bc = 10^n b_L c_L + 10^{n/2}(b_L c_R + b_R c_L) + b_R c_R$$

$\mathbf{4}$ recursive multiplications of number of size $n/2$ each plus 4 additions and left shifts (adding enough 0's to the right)

$$T(n) = 4T(n/2) + O(n) \qquad T(1) = O(1)$$

$T(n) = \Theta(n^2)$. No better than grade school multiplication!

11.3
Faster multiplication: Karatsuba's Algorithm

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 "Prince of Mathematicians"

Observation: Multiply two complex numbers: $(a+bi)$ and $(c+di)$

$$(a+bi)(c+di) = ac - bd + (ad+bc)i$$

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute $ac, bd, (a+b)(c+d)$. Then $(ad+bc) = (a+b)(c+d) - ac - bd$

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 "Prince of Mathematicians"

Observation: Multiply two complex numbers: $(a + bi)$ and $(c + di)$

$$(a + bi)(c + di) = ac - bd + (ad + bc)i$$

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute $ac, bd, (a + b)(c + d)$. Then $(ad + bc) = (a + b)(c + d) - ac - bd$

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 "Prince of Mathematicians"

Observation: Multiply two complex numbers: $(a + bi)$ and $(c + di)$

$$(a + bi)(c + di) = ac - bd + (ad + bc)i$$

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute $ac, bd, (a + b)(c + d)$. Then $(ad + bc) = (a + b)(c + d) - ac - bd$

Gauss technique for polynomials

$p(x) = ax + b$ and $q(x) = cx + d$.

$$p(x)q(x) = acx^2 + (ad + bc)x + bd.$$

$$p(x)q(x) = acx^2 + \big((a + b)(c + d) - ac - bd\big)x + bd.$$

Gauss technique for polynomials

$p(x) = ax + b$ and $q(x) = cx + d$.

$$p(x)q(x) = acx^2 + (ad + bc)x + bd.$$

$$p(x)q(x) = acx^2 + \big((a + b)(c + d) - ac - bd\big)x + bd.$$

Improving the Running Time

$$bc = b(x)c(x) = (b_L x + b_R)(c_L x + c_R)$$

Improving the Running Time

$$
\begin{aligned}
bc = b(x)c(x) &= (b_L x + b_R)(c_L x + c_R) \\
&= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R
\end{aligned}
$$

Improving the Running Time

$$\begin{aligned} bc &= b(x)c(x) = (b_L x + b_R)(c_L x + c_R) \\ &= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\ &= (b_L * c_L)x^2 + \Big((b_L + b_R) * (c_L + c_R) - b_L * c_L - b_R * c_R\Big)x + b_R * c_R \end{aligned}$$

Improving the Running Time

$$\begin{aligned} bc &= b(x)c(x) = (b_L x + b_R)(c_L x + c_R) \\ &= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\ &= (b_L * c_L)x^2 + \Big((b_L + b_R) * (c_L + c_R) - b_L * c_L - b_R * c_R\Big)x + b_R * c_R \end{aligned}$$

Recursively compute only $b_L c_L, b_R c_R, (b_L + b_R)(c_L + c_R)$.

Improving the Running Time

$$\begin{aligned} bc &= b(x)c(x) = (b_L x + b_R)(c_L x + c_R) \\ &= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\ &= (b_L * c_L)x^2 + \Big((b_L + b_R) * (c_L + c_R) - b_L * c_L - b_R * c_R\Big)x + b_R * c_R \end{aligned}$$

Recursively compute only $b_L c_L, b_R c_R, (b_L + b_R)(c_L + c_R)$.

Time Analysis

Running time is given by

$$T(n) = 3T(n/2) + O(n) \qquad T(1) = O(1)$$

which means $T(n) = O(n^{\log_2 3}) = O(n^{1.585})$

State of the Art

Schönhage-Strassen 1971: $O(n \log n \log \log n)$ time using Fast-Fourier-Transform (FFT)

Martin Fürer 2007: $O(n \log n 2^{O(\log^* n)})$ time

Conjecture

There is an $O(n \log n)$ time algorithm.

11.3.1
Solving the recurrences for fast multiplication

Analyzing the Recurrences

1. Basic divide and conquer: $T(n) = 4T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^2)$.
2. Saving a multiplication: $T(n) = 3T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^{1+\log 1.5})$

Use recursion tree method:

1. In both cases, depth of recursion $L = \log n$.
2. Work at depth i is $4^i n/2^i$ and $3^i n/2^i$ respectively: number of children at depth i times the work at each child
3. Total work is therefore $n \sum_{i=0}^{L} 2^i$ and $n \sum_{i=0}^{L} (3/2)^i$ respectively.

Analyzing the Recurrences

1. Basic divide and conquer: $T(n) = 4T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^2)$.
2. Saving a multiplication: $T(n) = 3T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^{1+\log 1.5})$

Use recursion tree method:

1. In both cases, depth of recursion $L = \log n$.
2. Work at depth i is $4^i n/2^i$ and $3^i n/2^i$ respectively: number of children at depth i times the work at each child
3. Total work is therefore $n\sum_{i=0}^{L} 2^i$ and $n\sum_{i=0}^{L}(3/2)^i$ respectively.

Analyzing the recurrence with four recursive calls

$T(n) = 4T(n/2) + O(n)$, $T(1) = 1$

Analyzing the recurrence with three recursive calls

$T(n) = 3T(n/2) + O(n)$, $T(1) = 1$

Analyzing the recurrence with two recursive calls

$T(n) = 2T(n/2) + O(n)$, $T(1) = 1$

11.4
Selecting in Unsorted Lists

11.4.1
Problem definition and basic algorithm

Rank of element in an array

A: an unsorted array of n integers

Definition 11.1.

For $1 \leq j \leq n$, element of rank j is the jth smallest element in A.

Unsorted array	16	14	34	20	12	5	3	19	11
Ranks	6	5	9	8	4	2	1	7	3

Sort of array	3	5	11	12	14	16	19	20	34

Problem - Selection

Input Unsorted array A of n integers **and** integer j

Goal Find the jth smallest number in A (rank j number)

Median: $j = \lfloor (n + 1)/2 \rfloor$

Simplifying assumption for sake of notation: elements of A are distinct

Problem - Selection

Input Unsorted array A of n integers **and** integer j

Goal Find the jth smallest number in A (rank j number)

Median: $j = \lfloor (n+1)/2 \rfloor$

Simplifying assumption for sake of notation: elements of A are distinct

Algorithm I

1. Sort the elements in A
2. Pick jth element in sorted order

Time taken $= O(n \log n)$

Do we need to sort? Is there an $O(n)$ time algorithm?

Algorithm I

1. Sort the elements in A
2. Pick jth element in sorted order

Time taken $= O(n \log n)$

Do we need to sort? Is there an $O(n)$ time algorithm?

Algorithm II

If j is small or $n - j$ is small then

1. Find j smallest/largest elements in A in $O(jn)$ time. (How?)
2. Time to find median is $O(n^2)$.

11.4.2
Quick select

QuickSelect

Divide and Conquer Approach

1. Pick a pivot element a from A
2. Partition A based on a.
   $A_{\text{less}} = \{x \in A \mid x \leq a\}$ and $A_{\text{greater}} = \{x \in A \mid x > a\}$
3. $|A_{\text{less}}| = j$: return a
4. $|A_{\text{less}}| > j$: recursively find jth smallest element in A_{less}
5. $|A_{\text{less}}| < j$: recursively find kth smallest element in A_{greater} where $k = j - |A_{\text{less}}|$.

Example

16	14	34	20	12	5	3	19	11

Time Analysis

1. Partitioning step: $O(n)$ time to scan A
2. How do we choose pivot? Recursive running time?

Suppose we always choose pivot to be $A[1]$.

Say A is sorted in increasing order and $j = n$.
Exercise: show that algorithm takes $\Omega(n^2)$ time

Time Analysis

1. Partitioning step: $O(n)$ time to scan A
2. How do we choose pivot? Recursive running time?

Suppose we always choose pivot to be $A[1]$.

Say A is sorted in increasing order and $j = n$.
Exercise: show that algorithm takes $\Omega(n^2)$ time

Time Analysis

1. Partitioning step: $O(n)$ time to scan A
2. How do we choose pivot? Recursive running time?

Suppose we always choose pivot to be $A[1]$.

Say A is sorted in increasing order and $j = n$.
Exercise: show that algorithm takes $\Omega(n^2)$ time

A Better Pivot

Suppose pivot is the ℓth smallest element where $n/4 \leq \ell \leq 3n/4$.
That is pivot is approximately in the middle of A
Then $n/4 \leq |A_{\text{less}}| \leq 3n/4$ and $n/4 \leq |A_{\text{greater}}| \leq 3n/4$. If we apply recursion,

$$T(n) \leq T(3n/4) + O(n)$$

Implies $T(n) = O(n)$!

How do we find such a pivot? Randomly? In fact works!
Analysis a little bit later.

Can we choose pivot deterministically?

A Better Pivot

Suppose pivot is the ℓth smallest element where $n/4 \leq \ell \leq 3n/4$.
That is pivot is approximately in the middle of A
Then $n/4 \leq |A_{\text{less}}| \leq 3n/4$ and $n/4 \leq |A_{\text{greater}}| \leq 3n/4$. If we apply recursion,

$$T(n) \leq T(3n/4) + O(n)$$

Implies $T(n) = O(n)$!

How do we find such a pivot? Randomly? In fact works!
Analysis a little bit later.

Can we choose pivot deterministically?

A Better Pivot

Suppose pivot is the ℓth smallest element where $n/4 \leq \ell \leq 3n/4$.
That is pivot is approximately in the middle of A
Then $n/4 \leq |A_{\text{less}}| \leq 3n/4$ and $n/4 \leq |A_{\text{greater}}| \leq 3n/4$. If we apply recursion,

$$T(n) \leq T(3n/4) + O(n)$$

Implies $T(n) = O(n)$!

How do we find such a pivot? Randomly? In fact works!
Analysis a little bit later.

Can we choose pivot deterministically?

A Better Pivot

Suppose pivot is the ℓth smallest element where $n/4 \le \ell \le 3n/4$.
That is pivot is approximately in the middle of A
Then $n/4 \le |A_{\text{less}}| \le 3n/4$ and $n/4 \le |A_{\text{greater}}| \le 3n/4$. If we apply recursion,

$$T(n) \le T(3n/4) + O(n)$$

Implies $T(n) = O(n)$!

How do we find such a pivot? Randomly? In fact works!
Analysis a little bit later.

Can we choose pivot deterministically?

A Better Pivot

Suppose pivot is the ℓth smallest element where $n/4 \leq \ell \leq 3n/4$.
That is pivot is approximately in the middle of A
Then $n/4 \leq |A_{\text{less}}| \leq 3n/4$ and $n/4 \leq |A_{\text{greater}}| \leq 3n/4$. If we apply recursion,

$$T(n) \leq T(3n/4) + O(n)$$

Implies $T(n) = O(n)$!

How do we find such a pivot? Randomly? In fact works!
Analysis a little bit later.

Can we choose pivot deterministically?

A Better Pivot

Suppose pivot is the ℓth smallest element where $n/4 \leq \ell \leq 3n/4$.
That is pivot is approximately in the middle of A
Then $n/4 \leq |A_{\text{less}}| \leq 3n/4$ and $n/4 \leq |A_{\text{greater}}| \leq 3n/4$. If we apply recursion,

$$T(n) \leq T(3n/4) + O(n)$$

Implies $T(n) = O(n)$!

How do we find such a pivot? Randomly? In fact works!
Analysis a little bit later.

Can we choose pivot deterministically?

11.4.3
Median of Medians

Divide and Conquer Approach

A game of medians

Idea

1. Break input A into many subarrays: $L_1, \ldots L_k$.
2. Find median m_i in each subarray L_i.
3. Find the median x of the medians $m_1, \ldots, m_k$.
4. Intuition: The median x should be close to being a good median of all the numbers in A.
5. Use x as pivot in previous algorithm.

New example

The input:

75	31	13	26	83	110	60	120	63	30	3	41	44	107	30	23	91	17	6	110
68	24	41	26	58	57	61	20	52	45	13	79	86	91	55	66	13	103	36	60
19	40	45	111	56	74	17	95	96	77	29	65	36	96	93	119	9	61	3	9
100	3	88	47	115	107	79	39	109	20	59	25	92	81	36	10	30	113	73	116
72	58	24	16	12	69	40	24	19	92	7	65	75	41	43	117	103	38	8	20

Compute median of the medians (recursive call):

72	74	13	66
31	60	65	30
41	39	75	61
26	63	91	8
58	45	43	60

After partition (pivot **60**):

19	3	13	16	12	57	17	20	19	20	3	25	92	109	96	79	110	69	83	75
41	24	24	26	56	17	40	24	52	30	7	60	77	81	63	61	107	115	111	72
20	31	41	26	58	30	60	39	36	45	13	65	75	91	120	66	74	61	88	68
9	40	45	47	3	13	23	55	30	44	29	65	86	96	95	117	91	103	100	110
36	58	8	6	38	9	10	43	41	36	59	79	92	107	93	119	103	113	73	116

Tail recursive call: Select element of rank **50** out of **56** elements.

19	3	13	16	12	57	17	20	19	20	3	25
41	24	24	26	56	17	40	24	52	30	7	
20	31	41	26	58	30	60	39	36	45	13	
9	40	45	47	3	13	23	55	30	44	29	
36	58	8	6	38	9	10	43	41	36	59	

New example

The input:

75	31	13	26	83	110	60	120	63	30	3	41	44	107	30	23	91	17	6	110
68	24	41	26	58	57	61	20	52	45	13	79	86	91	55	66	13	103	36	60
19	40	45	111	56	74	17	95	96	77	29	65	36	96	93	119	9	61	3	9
100	3	88	47	115	107	79	39	109	20	59	25	92	81	36	10	30	113	73	116
72	58	24	16	12	69	40	24	19	92	7	65	75	41	43	117	103	38	8	20

Compute median of the medians (recursive call):

72	74	13	66
31	60	65	30
41	39	75	61
26	63	91	8
58	45	43	60

After partition (pivot **60**):

19	3	13	16	12	57	17	20	19	20	3	25	92	109	96	79	110	69	83	75
41	24	24	26	56	17	40	24	52	30	7	60	77	81	63	61	107	115	111	72
20	31	41	26	58	30	60	39	36	45	13	65	75	91	120	66	74	61	88	68
9	40	45	47	3	13	23	55	30	44	29	65	86	96	95	117	91	103	100	110
36	58	8	6	38	9	10	43	41	36	59	79	92	107	93	119	103	113	73	116

Tail recursive call: Select element of rank **50** out of **56** elements.

19	3	13	16	12	57	17	20	19	20	3	25
41	24	24	26	56	17	40	24	52	30	7	
20	31	41	26	58	30	60	39	36	45	13	
9	40	45	47	3	13	23	55	30	44	29	
36	58	8	6	38	9	10	43	41	36	59	

New example

The input:

75	31	13	26	83	110	60	120	63	30	3	41	44	107	30	23	91	17	6	110
68	24	41	26	58	57	61	20	52	45	13	79	86	91	55	66	13	103	36	60
19	40	45	111	56	74	17	95	96	77	29	65	36	96	93	119	9	61	3	9
100	3	88	47	115	107	79	39	109	20	59	25	92	81	36	10	30	113	73	116
72	58	24	16	12	69	40	24	19	92	7	65	75	41	43	117	103	38	8	20

Compute median of the medians (recursive call):

72	74	13	66
31	60	65	30
41	39	75	61
26	63	91	8
58	45	43	60

After partition (pivot **60**):

19	3	13	16	12	57	17	20	19	20	3	25	92	109	96	79	110	69	83	75
41	24	24	26	56	17	40	24	52	30	7	60	77	81	63	61	107	115	111	72
20	31	41	26	58	30	60	39	36	45	13	65	75	91	120	66	74	61	88	68
9	40	45	47	3	13	23	55	30	44	29	65	86	96	95	117	91	103	100	110
36	58	8	6	38	9	10	43	41	36	59	79	92	107	93	119	103	113	73	116

Tail recursive call: Select element of rank **50** out of **56** elements.

19	3	13	16	12	57	17	20	19	20	3	25
41	24	24	26	56	17	40	24	52	30	7	
20	31	41	26	58	30	60	39	36	45	13	
9	40	45	47	3	13	23	55	30	44	29	
36	58	8	6	38	9	10	43	41	36	59	

New example

The input:

75	31	13	26	83	110	60	120	63	30	3	41	44	107	30	23	91	17	6	110
68	24	41	26	58	57	61	20	52	45	13	79	86	91	55	66	13	103	36	60
19	40	45	111	56	74	17	95	96	77	29	65	36	96	93	119	9	61	3	9
100	3	88	47	115	107	79	39	109	20	59	25	92	81	36	10	30	113	73	116
72	58	24	16	12	69	40	24	19	92	7	65	75	41	43	117	103	38	8	20

Compute median of the medians (recursive call):

72	74	13	66
31	60	65	30
41	39	75	61
26	63	91	8
58	45	43	60

After partition (pivot **60**):

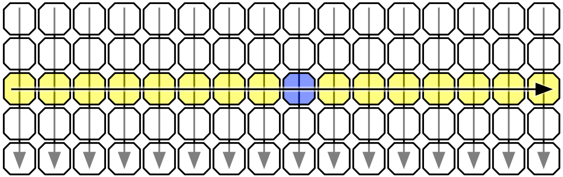
19	3	13	16	12	57	17	20	19	20	3	25	92	109	96	79	110	69	83	75
41	24	24	26	56	17	40	24	52	30	7	60	77	81	63	61	107	115	111	72
20	31	41	26	58	30	60	39	36	45	13	65	75	91	120	66	74	61	88	68
9	40	45	47	3	13	23	55	30	44	29	65	86	96	95	117	91	103	100	110
36	58	8	6	38	9	10	43	41	36	59	79	92	107	93	119	103	113	73	116

Tail recursive call: Select element of rank **50** out of **56** elements.

19	3	13	16	12	57	17	20	19	20	3	25
41	24	24	26	56	17	40	24	52	30	7	
20	31	41	26	58	30	60	39	36	45	13	
9	40	45	47	3	13	23	55	30	44	29	
36	58	8	6	38	9	10	43	41	36	59	

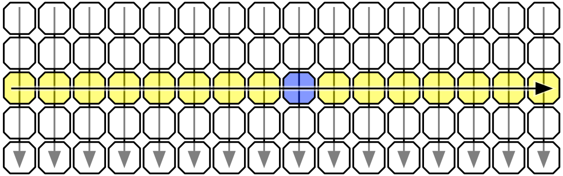
Example

11	7	3	42	174	310	1	92	87	12	19	15



Example

11	7	3	42	174	310	1	92	87	12	19	15



Choosing the pivot

A clash of medians

1. Partition array A into $\lceil n/5 \rceil$ lists of 5 items each.
   $L_1 = \{A[1], A[2], \ldots, A[5]\}$, $L_2 = \{A[6], \ldots, A[10]\}$, ...,
   $L_i = \{A[5i+1], \ldots, A[5i-4]\}$, ..., $L_{\lceil n/5 \rceil} = \{A[5\lceil n/5 \rceil - 4, \ldots, A[n]\}$.
2. For each i find median b_i of L_i using brute-force in $O(1)$ time. Total $O(n)$ time
3. Let $B = \{b_1, b_2, \ldots, b_{\lceil n/5 \rceil}\}$
4. Find median b of B

Lemma 11.2.

Median of B is an approximate median of A. That is, if b is used a pivot to partition A, then $|A_{less}| \leq 7n/10 + 6$ and $|A_{greater}| \leq 7n/10 + 6$.

Choosing the pivot

A clash of medians

1. Partition array A into $\lceil n/5 \rceil$ lists of 5 items each.
   $L_1 = \{A[1], A[2], \ldots, A[5]\}$, $L_2 = \{A[6], \ldots, A[10]\}$, $\ldots$,
   $L_i = \{A[5i+1], \ldots, A[5i-4]\}$, $\ldots$, $L_{\lceil n/5 \rceil} = \{A[5\lceil n/5 \rceil - 4, \ldots, A[n]\}$.
2. For each i find median b_i of L_i using brute-force in $O(1)$ time. Total $O(n)$ time
3. Let $B = \{b_1, b_2, \ldots, b_{\lceil n/5 \rceil}\}$
4. Find median b of B

Lemma 11.2.

Median of B is an approximate median of A. That is, if b is used a pivot to partition A, then $|A_{less}| \leq 7n/10 + 6$ and $|A_{greater}| \leq 7n/10 + 6$.

Algorithm for Selection

A storm of medians

```
select(A, j):
    Form lists L_1, L_2, ..., L_⌈n/5⌉ where L_i = {A[5i − 4], ..., A[5i]}
    Find median b_i of each L_i using brute-force
    Find median b of B = {b_1, b_2, ..., b_⌈n/5⌉}
    Partition A into A_less and A_greater using b as pivot
    if (|A_less|) = j return b
    else if (|A_less|) > j)
        return select(A_less, j)
    else
        return select(A_greater, j − |A_less|)
```

How do we find median of B?

Algorithm for Selection

A storm of medians

```
select(A, j):
    Form lists L_1, L_2, ..., L_⌈n/5⌉ where L_i = {A[5i − 4], ..., A[5i]}
    Find median b_i of each L_i using brute-force
    Find median b of B = {b_1, b_2, ..., b_⌈n/5⌉}
    Partition A into A_less and A_greater using b as pivot
    if (|A_less|) = j return b
    else if (|A_less|) > j)
        return select(A_less, j)
    else
        return select(A_greater, j − |A_less|)
```

How do we find median of B? Recursively!

Algorithm for Selection

A storm of medians

```
select(A, j):
    Form lists L_1, L_2, ..., L_{⌈n/5⌉} where L_i = {A[5i − 4], ..., A[5i]}
    Find median b_i of each L_i using brute-force
    Find median b of B = {b_1, b_2, ..., b_{⌈n/5⌉}}
    Partition A into A_less and A_greater using b as pivot
    if (|A_less|) = j return b
    else if (|A_less|) > j)
        return select(A_less, j)
    else
        return select(A_greater, j − |A_less|)
```

How do we find median of B? Recursively!

Algorithm for Selection

A storm of medians

select(A, j):

  Form lists $L_1, L_2, \ldots, L_{\lceil n/5 \rceil}$ where $L_i = \{A[5i-4], \ldots, A[5i]\}$

  Find median b_i of each L_i using brute-force

  $B = [b_1, b_2, \ldots, b_{\lceil n/5 \rceil}]$

  $b =$ **select**(B, $\lceil n/10 \rceil$)

  Partition A into A_{less} and A_{greater} using b as pivot

  if ($|A_{\text{less}}|) = j$ return b

  else if ($|A_{\text{less}}|) > j$)

    return **select**(A_{less}, j)

  else

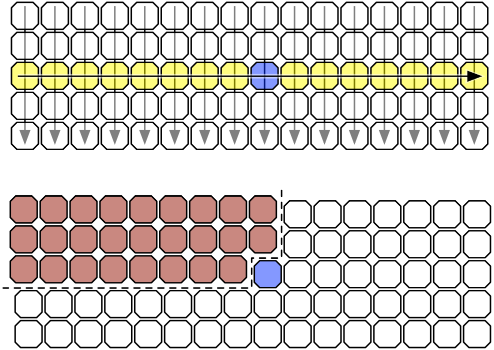
    return **select**(A_{greater}, $j - |A_{\text{less}}|$)

11.4.4
Median of medians is a good median

Median of Medians: Proof of Lemma

Proposition 11.3.

There are at least $3n/10 - 6$ *elements smaller than the median of medians* b.



Median of Medians: Proof of Lemma

Proposition 11.4.

There are at least $3n/10 - 6$ elements smaller than the median of medians b.

Proof.

At least half of the $\lfloor n/5 \rfloor$ groups have at least 3 elements smaller than b, except for the group containing b which has 2 elements smaller than b. Hence number of elements smaller than b is:

$$3\left\lfloor \frac{\lfloor n/5 \rfloor + 1}{2} \right\rfloor - 1 \geq 3n/10 - 6$$

□

Median of Medians: Proof of Lemma

Proposition 11.5.

There are at least $3n/10 - 6$ *elements smaller than the median of medians* b.

Corollary 11.6.

$|A_{greater}| \leq 7n/10 + 6$.

Via symmetric argument,

Corollary 11.7.

$|A_{less}| \leq 7n/10 + 6$.

11.4.5

Running time of deterministic median selection

Running time of deterministic median selection

A dance with recurrences

$$T(n) \leq T(\lceil n/5 \rceil) + \max\{T(|A_{\text{less}}|), T(|A_{\text{greater}})|\} + O(n)$$

From Lemma,

$$T(n) \leq T(\lceil n/5 \rceil) + T(\lfloor 7n/10 + 6 \rfloor) + O(n)$$

and

$$T(n) = O(1) \qquad n < 10$$

Exercise: show that $T(n) = O(n)$

Running time of deterministic median selection

A dance with recurrences

$$T(n) \leq T(\lceil n/5 \rceil) + \max\{T(|A_{\text{less}}|), T(|A_{\text{greater}})|\} + O(n)$$

From Lemma,

$$T(n) \leq T(\lceil n/5 \rceil) + T(\lfloor 7n/10 + 6 \rfloor) + O(n)$$

and

$$T(n) = O(1) \qquad n < 10$$

Exercise: show that $T(n) = O(n)$

Running time of deterministic median selection

A dance with recurrences

$$T(n) \leq T(\lceil n/5 \rceil) + \max\{T(|A_{\text{less}}|), T(|A_{\text{greater}})|\} + O(n)$$

From Lemma,

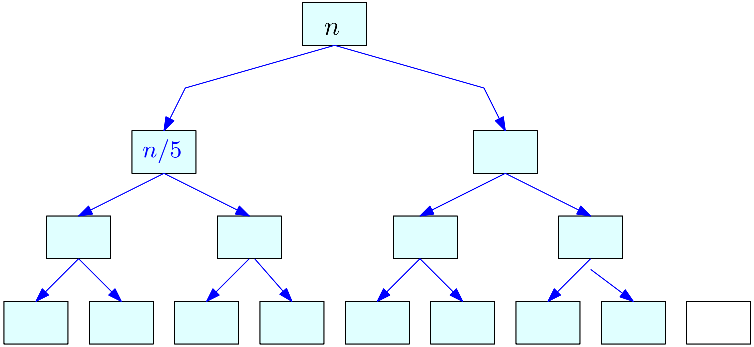
$$T(n) \leq T(\lceil n/5 \rceil) + T(\lfloor 7n/10 + 6 \rfloor) + O(n)$$

and

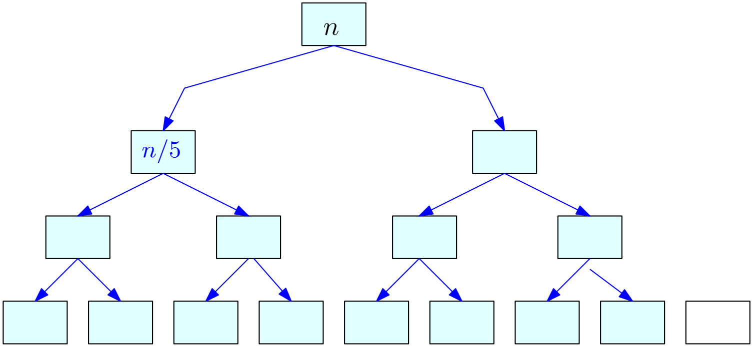
$$T(n) = O(1) \qquad n < 10$$

Exercise: show that $T(n) = O(n)$

Recursion tree fill in

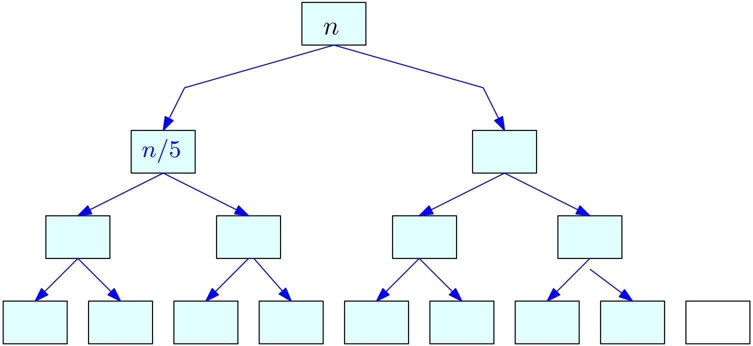


Recursion tree fill in



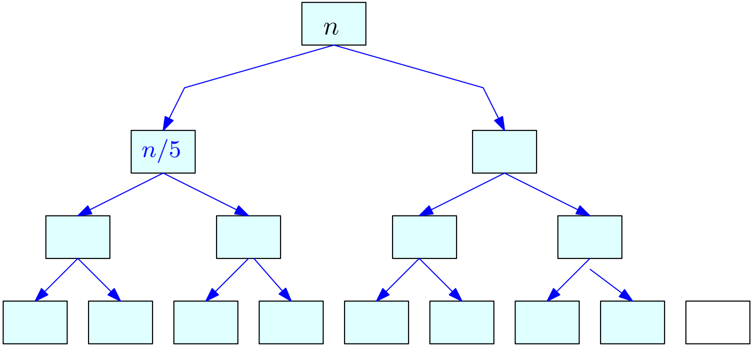
$(1/5)n$, $(7/10)n$

Recursion tree fill in



(1/25)*n*, (7/50)*n*, (7/50)*n*, (49/100)*n*

Recursion tree fill in



(1/125)*n*, (7/250)*n*, (7/250)*n*, (49/500)*n*, (7/250)*n*, (49/500)*n*, (49/500)*n*, (343/1000)*n*

11.4.6
Epilogue: On selection in linear time

Summary: Selection in linear time

Theorem 11.8.

The algorithm **select**$(A[1 .. n], k)$ *computes in* $O(n)$ *deterministic time the* k*th smallest element in* A.

On the other hand, we have:

Lemma 11.9.

The algorithm **QuickSelect**$(A[1 .. n], k)$ *computes the* k*th smallest element in* A. *The running time of* **QuickSelect** *is* $\Theta(n^2)$ *in the worst case.*

Questions to ponder

1. Why did we choose lists of size **5**? Will lists of size **3** work?
2. Write a recurrence to analyze the algorithm's running time if we choose a list of size k.

Median of Medians Algorithm

Due to:
M. Blum, R. Floyd, D. Knuth, V. Pratt, R. Rivest, and R. Tarjan.
"Time bounds for selection".
Journal of Computer System Sciences (JCSS), 1973.

How many Turing Award winners in the author list?
All except Vaughn Pratt!

Median of Medians Algorithm

Due to:
M. Blum, R. Floyd, D. Knuth, V. Pratt, R. Rivest, and R. Tarjan.
"Time bounds for selection".
Journal of Computer System Sciences (JCSS), 1973.

How many Turing Award winners in the author list?
All except Vaughn Pratt!

Median of Medians Algorithm

Due to:
M. Blum, R. Floyd, D. Knuth, V. Pratt, R. Rivest, and R. Tarjan.
"Time bounds for selection".
Journal of Computer System Sciences (JCSS), 1973.

How many Turing Award winners in the author list?
All except Vaughn Pratt!

Takeaway Points

1. Recursion tree method and guess and verify are the most reliable methods to analyze recursions in algorithms.
2. Recursive algorithms naturally lead to recurrences.
3. Some times one can look for certain type of recursive algorithms (reverse engineering) by understanding recurrences and their behavior.