Kartsuba's Algorithm and Linear Time Selection

Lecture 11

February 23, 2023

Part I

Fast Multiplication

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: $\Theta(n)$
2. Number of partial products: $\Theta(n)$
3. Addition of partial products: $\Theta(n^2)$
4. Total time: $\Theta(n^2)$

A Trick of Gauss

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

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

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

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 “Prince of Mathematicians”

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

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

How many multiplications do we need?

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 “Prince of Mathematicians”

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

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. $x = x_{n-1}x_{n-2}\ldots x_0$ and $y = y_{n-1}y_{n-2}\ldots y_0$
2. $x = x_{n-1}\ldots x_{n/2}0\ldots 0 + x_{n/2-1}\ldots x_0$

$$19287713 = 19280000 + 7713$$

3. $x_L = x_{n-1}\ldots x_{n/2}$ and $x_R = x_{n/2-1}\ldots x_0$ and $x = 10^{n/2}x_L + x_R$

$$19287713 = 10^4 \times 1928 + 7713$$

4. Similarly $y = 10^{n/2}y_L + y_R$ where $y_L = y_{n-1}\ldots y_{n/2}$ and $y_R = y_{n/2-1}\ldots y_0$

Example

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

Divide and Conquer

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

1. $x = x_{n-1}x_{n-2}\dots x_0$ and $y = y_{n-1}y_{n-2}\dots y_0$
2. $x = 10^{n/2}x_L + x_R$ where $x_L = x_{n-1}\dots x_{n/2}$ and $x_R = x_{n/2-1}\dots x_0$
3. $y = 10^{n/2}y_L + y_R$ where $y_L = y_{n-1}\dots y_{n/2}$ and $y_R = y_{n/2-1}\dots y_0$

Therefore

$$\begin{aligned} xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\ &= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

Time Analysis

$$\begin{aligned} xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\ &= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

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

Time Analysis

$$
\begin{aligned}
xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\
&= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R
\end{aligned}
$$

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

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

Time Analysis

$$
\begin{aligned}
xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\
&= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R
\end{aligned}
$$

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

$$xy = (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R)$$
$$= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

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!

Can we invoke Gauss's trick here?

Improving the Running Time

$$\begin{aligned} xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\ &= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

Gauss trick: $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Improving the Running Time

$$
\begin{aligned}
xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\
&= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R
\end{aligned}
$$

Gauss trick: $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Recursively compute only $x_L y_L, x_R y_R, (x_L + x_R)(y_L + y_R)$.

Improving the Running Time

$$xy = (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R)$$
$$= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Gauss trick: $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Recursively compute only $x_L y_L, x_R y_R, (x_L + x_R)(y_L + y_R)$.

Time Analysis

Running time is given by

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

which means

Improving the Running Time

$$\begin{aligned} xy &= (10^{n/2}x_L + x_R)(10^{n/2}y_L + y_R) \\ &= 10^n x_L y_L + 10^{n/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

Gauss trick: $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Recursively compute only $x_L y_L, x_R y_R, (x_L + x_R)(y_L + y_R)$.

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)

Conjecture[Schönhage & Strassen 1971]

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

State of the Art

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

Conjecture[Schönhage & Strassen 1971]

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

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

State of the Art

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

Conjecture[Schönhage & Strassen 1971]

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

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

Theorem (Harvey and van der Hoeven, Annals of Math, 2021)

Integer multiplication can be done in $O(n \log n)$ time.

Open problem: Is there an $O(n)$ time algorithm? Seems implausible but lower bounds are very hard!

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:** $T(n) = \Theta(n^{1+\log 1.5})$

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.

Recursion tree analysis

Part II

Selecting in Unsorted Lists

Rank of element in an array

A: an unsorted array of n integers

Definition

For $1 \leq j \leq n$, element of rank j is the j'th 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: $j = \lfloor (n+1)/2 \rfloor$

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: elements of A are distinct

Caveat: simplifying assumptions useful for thinking and exposition but need to be very careful when finalizing details, especially when translating into code/implementations

Algorithm I

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

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

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 $O(n^2)$.

Divide and Conquer Approach

1. Pick a pivot element a from A
2. Partition A based on a.
   $A_{\text{less}} = \{x \in A \mid x \le a\}$ 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?

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 $A[1]$.

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,

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

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?

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?

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 in CS 473 or in other courses that cover randomized algorithms

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 in CS 473 or in other courses that cover randomized algorithms

Can we choose pivot deterministically?

Divide and Conquer Approach

A game of medians

Idea

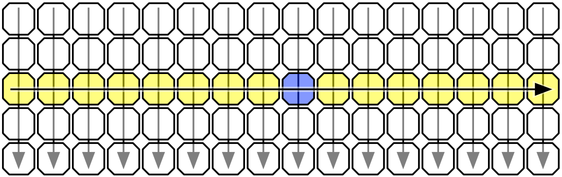
1. Break input A into many subarrays: $L_1, \ldots L_k$.
2. Find median m_i in each subarray L_i.
3. Find the median x of the medians $m_1, \ldots, m_k$.
4. Intuition: The median x should be close to being a good median of all the numbers in A.
5. Use x as pivot in previous algorithm.

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.
   $L_1 = \{A[1], A[2], \ldots, A[5]\}$, $L_2 = \{A[6], \ldots, A[10]\}$, $\ldots$,
   $L_i = \{A[5i+1], \ldots, A[5i-4]\}$, $\ldots$,
   $L_{\lceil n/5 \rceil} = \{A[5\lceil n/5 \rceil - 4, \ldots, A[n]\}$.
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

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

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|)
```

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 $L_1, L_2, \ldots, L_{\lceil n/5 \rceil}$ 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}, $j - |A_{\text{less}}|$)

Running time of deterministic median selection

A dance with recurrences

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

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

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

Running time of deterministic median selection

A dance with recurrences

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

From Lemma,

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

and

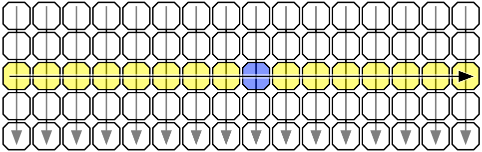
$$T(n) = O(1) \qquad n < 10$$

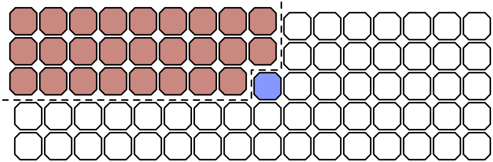
Exercise: show that $T(n) = O(n)$

Median of Medians: Proof of Lemma

Proposition

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





Median of Medians: Proof of Lemma

Proposition

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:

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

□

Median of Medians: Proof of Lemma

Proposition

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

Corollary

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

Via symmetric argument,

Corollary

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

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.

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?

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.