Recursion

Integer Multiplication

Integer Multiplication

$$x \cdot y = ?$$

CPU/GPU:
A basic operation

You and me:
Practicing since elementary school…

Outline

- School algorithm
  - $O(n^2)$ time
- Divide and Conquer
  - Naive: $O(n^2)$ time
  - Karatsuba method: $O\left(n^{\log 3}\right) = O(n^{1.585})$

Learning objectives:

You are able to

- Derive the runtime recurrences of the naïve and Karatsuba algorithms for multiplication
- Solve the runtime recurrence for Karatsuba's algorithms
- Name the state of the art (SOTA) runtime for multiplication

The Lattice Algorithm

Assume 10-ary digits.
A number is an array of digits, starting with the least significant digit.

Integer: 8343

Index	**4**	**3**	**2**	**1**	**0**
Value		8	3	4	3

The Lattice Algorithm

Multiplication in school:
Multiply n-digit number x with a single digit number y

$$x \cdot y = \Sigma_{0 \le i < n} 10^i \cdot x_i \cdot y$$

The Lattice Algorithm

Multiplication in school:
Multiply n-digit number x with a single digit number y

$$x \cdot y = \Sigma_{0 \le i < n} 10^i \cdot x_i \cdot y$$

x_i is the entry at index i. Multiply two single digit numbers.

$$\begin{array}{rr}
 & 3285 \\
\cdot & 4 \\
\hline
 & 20 \\
 & 320 \\
 & 800 \\
+ & 12000 \\
\hline
 & 13140
\end{array}$$

The Lattice Algorithm

Multiplication in school:
Multiply n-digit number x with a single digit number y

$$x \cdot y = \Sigma_{0 \leq i < n} 10^i \cdot x_i \cdot y$$

$$\begin{array}{rr} & 3285 \\ \cdot & 4 \\ \hline & 20 \\ & 320 \\ & 800 \\ + & 12000 \\ \hline & 13140 \end{array}$$

Add the “hold” and the current product

The Lattice Algorithm

Multiplication in school:
Multiply n-digit number x with a single digit number y

$$x \cdot y = \Sigma_{0 \le i < n} 10^i \cdot x_i \cdot y$$

$$\begin{array}{r} 3285 \\ \cdot \quad\quad 4 \\ \hline 20 \\ 320 \\ 800 \\ +\ 12000 \\ \hline 13140 \end{array}$$

$O(n)$ time

Add the “hold” and the current product

The Lattice Algorithm

Multiplication in school:
Multiply n-digit number x with an m-digit number y

$$x \cdot y = \Sigma_{0 \le j < m} \Sigma_{0 \le i < n} \, 10^i \cdot x_i \cdot 10^j \cdot y_j$$

$$\begin{array}{rr}
 & 3285 \\
\cdot & 457 \\
\hline
 & 22995 \\
 & 164250 \\
+ & 1314000 \\
\hline
 & 1501245
\end{array}$$

The Lattice Algorithm

Multiplication in school:
Multiply n-digit number x with an m-digit number y

$$x \cdot y = \Sigma_{0 \le j < m} \Sigma_{0 \le i < n} \, 10^i \cdot x_i \cdot 10^j \cdot y_j$$

$$\begin{array}{rr} & 3285 \\ \cdot & 457 \\ \hline & 22995 \\ & 164250 \\ + & 1314000 \\ \hline & 1501245 \end{array}$$

$O(n \cdot m)$ time

Outline

- School algorithm
  - $O(n^2)$ time

- Divide and Conquer
  - Naive: $O(n^2)$ time
  - Karatsuba method: $O\left(n^{\log 3}\right) = O(n^{1.585})$

Divide and Conquer

Observation:
If the number system is base 10, multiplying with a power of ten is easy. Just add zeros.

$$10^5 \cdot 12345 = 12345\ 00000$$

Divide and Conquer

Observation:
If the number system is base 10, multiplying with a power of ten is easy. Just add zeros.

$$10^5 \cdot 12345 = 12345\ 00000$$

Shift the array by 5

Index	**4**	**3**	**2**	**1**	**0**
Value	1	2	3	4	5

Divide and Conquer

Observation:
If the number system is base 10, multiplying with a power of ten is easy. Just add zeros.

$$10^5 \cdot 12345 = 12345\,00000$$

Shift the array by 5

Index	**4**	**3**	**2**	**1**	**0**
Value	1	2	3	4	5

Index	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Value	1	2	3	4	5	0	0	0	0	0

Divide and Conquer

Observation:
If the number system is base 10, multiplying with a power of ten is easy. Just add zeros.

$O(n)$ time for an array of n digits

$$10^5 \cdot 12345 = 12345\ 00000$$

Shift the array by 5

Index	**4**	**3**	**2**	**1**	**0**
Value	1	2	3	4	5

Index	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Value	1	2	3	4	5	0	0	0	0	0

Divide and Conquer

Divide:

$$1234 = 10^2 \cdot 12 + 34$$

Re-write $\boldsymbol{x \cdot y}$:

Suppose x and y are n digit numbers. Write

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

Divide and Conquer

Divide:

$$1234 = 10^2 \cdot 12 + 34$$

Re-write $\boldsymbol{x \cdot y}$:

Suppose x and y are n digit numbers. Write

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

$\boldsymbol{n}$-digit multiplication:

$$\begin{aligned} x \cdot y &= \left(10^{n/2} \cdot a + b\right) \cdot \left(10^{n/2} \cdot c + d\right) \\ &= 10^n \cdot ac + 10^{n/2} \cdot (bc + ad) + bd \end{aligned}$$

Divide and Conquer

Divide:

$$1234 = 10^2 \cdot 12 + 34$$

Re-write $\boldsymbol{x \cdot y}$:

Suppose x and y are n digit numbers. Write

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

$\boldsymbol{n}$-digit multiplication:

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

$(n/2)$-digit multiplications!

Divide and Conquer

Runtime recurrence:

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

Additions and shifting

$\boldsymbol{n}$-digit multiplication:

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

$(n/2)$-digit multiplications!

Divide and Conquer

Runtime recurrence:

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

Divide and Conquer

Runtime recurrence:

$$T(n) = 4 \cdot T(n/2) + O(n)$$
$$= 4^{\log_2 n} \cdot O(1) + O(n \log n)$$

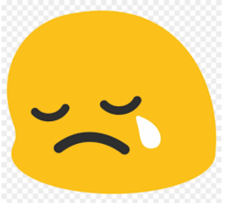
Master theorem

Divide and Conquer

Runtime recurrence:

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) + O(n) \\
&= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\
&= O(n^2) + O(n \log n) = O(n^2)
\end{aligned}
$$

Not better than the lattice algorithm



Outline

- School algorithm
  - $O(n^2)$ time

- Divide and Conquer
  - Naive: $O(n^2)$ time
  - Karatsuba method: $O\left(n^{\log 3}\right) = O(n^{1.585})$

Karatsuba's Method

Problem:
4 multiplications per recursion is too much

Runtime recurrence:

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) + O(n) \\
&= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\
&= O(n^2) + O(n \log n) = O(n^2)
\end{aligned}
$$

Karatsuba's Method

Problem:
4 multiplications per recursion is too much

Runtime recurrence:

$$\begin{aligned} T(n) &= 4 \cdot T(n/2) + O(n) \\ &= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\ &= O(n^2) + O(n \log n) = O(n^2) \end{aligned}$$

Karatsuba:
3 multiplications is enough!

Karatsuba's Method

Problem:
4 multiplications per recursion is too much

Runtime recurrence:

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) + O(n) \\
&= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\
&= O(n^2) + O(n \log n) = O(n^2)
\end{aligned}
$$

Karatsuba:
3 multiplications is enough!

Runtime recurrence:

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

Karatsuba's Method

Problem:
4 multiplications per recursion is too much

Runtime recurrence:

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) + O(n) \\
&= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\
&= O(n^2) + O(n \log n) = O(n^2)
\end{aligned}
$$

Karatsuba:
3 multiplications is enough!

Runtime recurrence:

$$
\begin{aligned}
T(n) &= 3 \cdot T(n/2) + O(n) \\
&= 3^{\log_2 n} \cdot O(1) + O(n \log n)
\end{aligned}
$$

Master theorem

Karatsuba's Method

Problem:
4 multiplications per recursion is too much

Runtime recurrence:

$$\begin{aligned} T(n) &= 4 \cdot T(n/2) + O(n) \\ &= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\ &= O(n^2) + O(n \log n) = O(n^2) \end{aligned}$$

Karatsuba:
3 multiplications is enough!

Runtime recurrence:

$$\begin{aligned} T(n) &= 3 \cdot T(n/2) + O(n) \\ &= 3^{\log_2 n} \cdot O(1) + O(n \log n) \\ &= O\left(n^{\log 3}\right) + O(n \log n) \\ &= O(n^{1.585}) \end{aligned}$$

Karatsuba's Method

Problem:
4 multiplications per recursion is too much

Runtime recurrence:

$$\begin{aligned} T(n) &= 4 \cdot T(n/2) + O(n) \\ &= 4^{\log_2 n} \cdot O(1) + O(n \log n) \\ &= O(n^2) + O(n \log n) = O(n^2) \end{aligned}$$

Karatsuba:
3 multiplications is enough!

Runtime recurrence:

$$\begin{aligned} T(n) &= 3 \cdot T(n/2) + O(n) \\ &= 3^{\log_2 n} \cdot O(1) + O(n \log n) \\ &= O\left(n^{\log 3}\right) + O(n \log n) \\ &= O(n^{1.585}) \end{aligned}$$

$$3 = 2^{\log 3}$$

$$(x^a)^b = \left(x^b\right)^a$$

Karatsuba's Method

Karatsuba:
3 multiplications
is enough!

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

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

Karatsuba's Method

Karatsuba:
3 multiplications
is enough!

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

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

Algebraic identity:

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

Karatsuba's Method

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

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

Algebraic identity:

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

Multiply(x, y, n):
$m \coloneqq n/2$
$ac =$ **Multiply(**a, c, m**)**
$bd =$ **Multiply(**b, d, m**)**
$f \coloneqq a - b$
$g \coloneqq c - d$
$ad + bc = ac + bd -$ **Multiply(**f, g, m**)**
Return $10^{2m} \cdot ac + 10^{m} \cdot (ad + bc) + bd$

Karatsuba's Method

n-bit integers
For simplicity, powers of 2

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

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

Multiply(x, y, n)**:**
$m \coloneqq n/2$
$ac =$ **Multiply**(a, c, m)
$bd =$ **Multiply**(b, d, m)
$f \coloneqq a - b$
$g \coloneqq c - d$
$ad + bc = ac + bd -$ **Multiply**(f, g, m)
Return $10^{2m} \cdot ac + 10^{m} \cdot (ad + bc) + bd$

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

Karatsuba's Method

n-bit integers
For simplicity, powers of 2

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

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

Algebraic identity:

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

Multiply(x, y, n):

$m \coloneqq n/2$
ac = **Multiply(a, c, m)**
bd = **Multiply(b, d, m)**
$f \coloneqq a - b$
$g \coloneqq c - d$
$ad + bc = ac + bd -$ **Multiply(f, g, m)**
Return $10^{2m} \cdot ac + 10^m \cdot (ad + bc) + bd$

Karatsuba's Method

n-bit integers
For simplicity, powers of 2

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

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

Multiply(x, y, n):
$m \coloneqq n/2$
ac = **Multiply**(a, c, m)
bd = **Multiply**(b, d, m)
$f \coloneqq a - b$
$g \coloneqq c - d$
$ad + bc = ac + bd -$ **Multiply**(f, g, m)
Return $10^{2m} \cdot ac + 10^m \cdot (ad + bc) + bd$

Algebraic identity:

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

Karatsuba's Method

n-bit integers
For simplicity, powers of 2

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

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

Multiply(x, y, n):
$m := n/2$
ac = **Multiply(a, c, m)**
bd = **Multiply(b, d, m)**
$f := a - b$
$g := c - d$
$ad + bc = ac + bd -$ **Multiply(f, g, m)**
Return $10^{2m} \cdot ac + 10^{m} \cdot (ad + bc) + bd$

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

3 recursive calls!

Karatsuba's Method

Problem:
Naïve recursive algorithm splits to two but needs 4 recursive calls

Runtime recurrence:

$$T(n) = 4 \cdot T(n/2) + O(n)$$
$$= O(n^2)$$

Karatsuba:
3 multiplications is enough!

Algebraic trick

Runtime recurrence:

$$T(n) = 3 \cdot T(n/2) + O(n)$$
$$= O(n^{1.585})$$

Progress on Integer Multiplication

$O(n^2)$	- History
$O(n^{1.585})$	- Karatsuba 1962
$O(n \log n \log \log n)$	- Schönhage & Strassen 1971
$O\left(n \log n \cdot 2^{O(\log^* n)}\right)$	- Fürer 2007
$O(n \log n)$	- Harvey & van der Hoeven 2019