

Introduction to Algorithms

Divide and Conquer Multiplication

Integer Multiplication

Integer Arithmetic

Add: Given two n -bit integers a and b , compute $a + b$.

Add

$O(n)$ bit operations.

Multiply: Given two n -bit integers a and b , compute $a \times b$.
The “grade school” method:

$O(n^2)$ bit operations.

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
<hr/>								
1	0	1	0	1	0	0	1	0

[illegible]

Multiply

How to use Divide and Conquer?

Suppose we want to multiply two 2-digit integers (32,45).

We can do this by multiplying four 1-digit integers

Then, use add/shift to obtain the result:

$$x = 10x_1 + x_0$$

$$y = 10y_1 + y_0$$

$$xy = (10x_1 + x_0)(10y_1 + y_0)$$

$$= 100 x_1y_1 + 10(x_1y_0 + x_0y_1) + x_0y_0$$

4	5	$y_1 y_0$
3	2	$x_1 x_0$
<hr/>		
1	0	$x_0 \cdot y_0$
0	8	$x_0 \cdot y_1$
1	5	$x_1 \cdot y_0$
1	2	$x_1 \cdot y_1$
<hr/>		
1	4	4 0

Same idea works when multiplying n-digit integers:

- Divide into 4 n/2-digit integers.
- Recursively multiply
- Then merge solutions

A Divide and Conquer for Integer Multiplication

Let x, y be two n -bit integers

Write $x = 2^{n/2}x_1 + x_0$ and $y = 2^{n/2}y_1 + y_0$

where x_0, x_1, y_0, y_1 are all $n/2$ -bit integers.

$$x = 2^{n/2} \cdot x_1 + x_0$$

$$y = 2^{n/2} \cdot y_1 + y_0$$

$$\begin{aligned} xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \\ &= 2^n \cdot x_1y_1 + 2^{n/2} \cdot (x_1y_0 + x_0y_1) + x_0y_0 \end{aligned}$$

Therefore,

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$$

So,

$$T(n) = \Theta(n^2).$$

We only need 3 values
 $x_1y_1, x_0y_0, x_1y_0 + x_0y_1$
Can we find all 3 by only
3 multiplication?

$$T(n) = a T\left(\frac{n}{b}\right) + cn^k \quad \text{If } a > b^k \text{ then } T(n) = \Theta(n^{\log_b a})$$

Key Trick: 4 multiplies at the price of 3

$$x = 2^{n/2} \cdot x_1 + x_0$$

$$y = 2^{n/2} \cdot y_1 + y_0$$

$$\begin{aligned} xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \\ &= 2^n \cdot x_1 y_1 + 2^{n/2} (x_1 y_0 + x_0 y_1) + x_0 y_0 \end{aligned}$$

$$\alpha = x_1 + x_0$$

$$\beta = y_1 + y_0$$

$$\alpha\beta = (x_1 + x_0)(y_1 + y_0)$$

$$= x_1 y_1 + (x_1 y_0 + x_0 y_1) + x_0 y_0$$

$$(x_1 y_0 + x_0 y_1) = \alpha\beta - x_1 y_1 - x_0 y_0$$

Key Trick: 4 multiplies at the price of 3

Theorem [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in $O(n^{1.585...})$ bit operations.

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \Rightarrow \alpha = x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \Rightarrow \beta = y_1 + y_0 \\xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \\&= \underbrace{2^n \cdot x_1 y_1}_A + \underbrace{2^{n/2} \cdot (x_1 y_0 + x_0 y_1)}_{\alpha\beta - A - B} + \underbrace{x_0 y_0}_B\end{aligned}$$

To multiply two n-bit integers:

Add two n/2 bit integers.

Multiply **three** n/2-bit integers.

Add, subtract, and shift n/2-bit integers to obtain result.

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O\left(n^{\log_2 3}\right) = O(n^{1.585...})$$

Integer Multiplication (Summary)

- Naïve: $\Theta(n^2)$
- Karatsuba: $\Theta(n^{1.585\dots})$
- **Amusing exercise:** generalize Karatsuba to do 5 size $n/3$ subproblems
This gives $\Theta(n^{1.46\dots})$ time algorithm
- Best known algorithm runs in $\Theta(n \log n)$ using fast Fourier transform
but mostly unused in practice (unless you need really big numbers - a billion digits of π , say)
- Best lower bound $\Omega(n)$: A fundamental open problem

D&C Summary

Idea:

“Two halves are better than a whole”

- if the base algorithm has super-linear complexity.

“If a little's good, then more's better”

- repeat above, recursively
- Applications: Many.
 - Binary Search, Merge Sort, (Quicksort),
 - Root of a Function
 - Closest points,
 - Integer multiplication
 - Median
 - Matrix Multiplication