Integer Multiplication

David Fernández-Baca

Computer Science 311
Iowa State University

February 6, 2022

Integer Addition

Addition: Given two n-bit integers x and y, compute $x + y$.
Subtraction: Given two n-bit integers x and y, compute $x - y$.
Grade-school algorithm: $\Theta(n)$ 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
1	0	1	0	1	0	0	1	0

Source: W

Integer Multiplication

Integer multiplication

Multiplication: Given two n-bit integers x and y, compute $x \times y$.

Grade-school algorithm (long multiplication): $\Theta(n^2)$ bit operations.

								1	1	0	1	0	1	0	1
							×	0	1	1	1	1	1	0	1
								1	1	0	1	0	1	0	1
							0	0	0	0	0	0	0	0	
						1	1	0	1	0	1	0	1		
					1	1	0	1	0	1	0	1			
				1	1	0	1	0	1	0	1				
			1	1	0	1	0	1	0	1					
		1	1	0	1	0	1	0	1						
	0	0	0	0	0	0	0	0							
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1

Source: W

Divide-and-Conquer Multiplication

$$m = \lceil n/2 \rceil$$

$$a = \lfloor x/2^m \rfloor \qquad b = x \bmod 2^m$$

$$c = \lfloor y/2^m \rfloor \qquad d = y \bmod 2^m$$

$$x = \underbrace{1\,0\,0\,0}_{a}\;\underbrace{1\,1\,0\,1}_{b}$$

$$y = \underbrace{1\,1\,1\,0}_{c}\;\underbrace{0\,0\,0\,1}_{d}$$

Source: W

Need 4 multiplications of $n/2$-bit integers to compute $x \times y$:

$$x \times y = (2^m a + b)(2^m c + d) = 2^{2m} \underset{1}{ac} + 2^m (\underset{2}{bc} + \underset{3}{ad}) + \underset{4}{bd}$$

```
Multiply(x, y, n):
    if n == 1 then
        return x × y
    else
        m = ⌈n/2⌉
        a = ⌊x/2^m⌋; b = x mod 2^m
        c = ⌊y/2^m⌋; d = y mod 2^m
        e = Multiply(a, c, m)
        f = Multiply(b, d, m)
        g = Multiply(b, c, m)
        h = Multiply(a, d, m)
        return 2^{2m} e + 2^m (g + h) + f
```

Proposition

`Multiply` *takes* $\Theta(n^2)$ *time.*

Proof.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 4T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1. \end{cases}$$

- Apply Master Theorem, with

$$a = 4, \quad b = 2, \quad \log_b a = \log_2 4 = 2, \quad f(n) = cn.$$

- $n^{\log_b a} = n^2$, so $f(n) = cn = O(n^{\log_b a - \epsilon})$, for $\epsilon > 0$.
- Case 1 of Master Theorem applies:

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2).$$

.

□

`Multiply`: Summary

- `Multiply` runs in $\Theta(n^2)$ time — no better than grade school algorithm.
- Reason: `Multiply` creates too many subproblems.
  - Number of nodes in recursion tree grows by a factor of 4 from one level to next.
- We can reduce the running time from $\Theta(n^2)$ to $\Theta(n^{1.585})$ by reducing the number of subproblems from 4 to 3, while the time to divide and combine stays the same: $\Theta(n)$.
  - Karatsuba's algorithm — next section.

Karatsuba's Algorithm

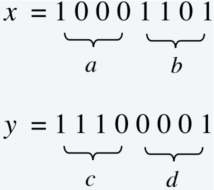
Karatsuba's Trick

Key identity:

$$bc + ad = \underbrace{ac}_{1} + \underbrace{bd}_{2} - \underbrace{(a-b)(c-d)}_{3}).$$

As before,

$$\begin{aligned} m &= \lceil n/2 \rceil \\ a &= \lfloor x/2^m \rfloor & \qquad b &= x \bmod 2^m \\ c &= \lfloor y/2^m \rfloor & \qquad d &= y \bmod 2^m \end{aligned}$$



Source: W

Now we only need 3 multiplications of $n/2$-bit integers.

$$\begin{aligned} x \times y = (2^m a + b)(2^m c + d) &= 2^{2m} ac + 2^m (bc + ad) + bd \\ &= 2^{2m} \underbrace{ac}_{1} + 2^m (\underbrace{ac}_{1} + \underbrace{bd}_{2} - \underbrace{(a-b)(c-d)}_{3}) + \underbrace{bd}_{2} \end{aligned}$$

```
KaratsubaMultiply(x, y, n):
    if n == 1 then
        return x × y
    else
        m = ⌈n/2⌉
        a = ⌊x/2^m⌋; b = x mod 2^m
        c = ⌊y/2^m⌋; d = y mod 2^m
        e = KaratsubaMultiply(a, c, m)
        f = KaratsubaMultiply(b, d, m)
        g = KaratsubaMultiply(|a − b|, |c − d|, m)
        Flip sign of g if needed
        return 2^{2m} e + 2^m (e + f − g) + f
```

Proposition

`KaratsubaMultiply` *takes* $\Theta(n^{\log_2 3})$ *time.*

Proof.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 3T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1. \end{cases}$$

- Apply Master Theorem, with

$$a = 3, \quad b = 2, \quad \log_b a = \log_2 3 \approx 1.585, \quad f(n) = cn.$$

- $f(n) = cn = O(n^{\log_b a - \epsilon}) = O(n^{1.585-\epsilon})$, for $\epsilon > 0$.
- Case 1 of Master Theorem applies:

$$T(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.585}).$$

□

Bibliography

Reference

[KT] Jon Kleinberg and Éva Tardos, *Algorithm Design*, Addison-Wesley, 2006.

[W] Kevin Wayne, *Lecture Slides for Algorithm Design by Jon Kleinberg and Éva Tardos*, `http://www.cs.princeton.edu/~wayne/kleinberg-tardos/`.

[WKA] Wikipedia. *Karatsuba Algorithm*, `https://en.wikipedia.org/wiki/Karatsuba_algorithm`.