

Divide and Conquer: Multiplication

CMSC 451, Summer 2009

Announcements / Reminders

- Late homework due now
- Homework 2 due next week:
 - 5.3, 5.5
 - Algorithm Write-Up Guidelines reminder
 - Group names on the paper

Closest Pair Algorithm

```

Closest-Pair( $p_1, \dots, p_n$ ) {
  Compute separation line L such that half the points
  are on one side and half on the other side.  $O(n \log n)$ 
   $\delta_1$  = Closest-Pair(left half)  $2T(n/2)$ 
   $\delta_2$  = Closest-Pair(right half)
   $\delta$  =  $\min(\delta_1, \delta_2)$ 
  Delete all points further than  $\delta$  from separation
  line L  $O(n)$ 
  Sort remaining points by y-coordinate.  $O(n \log n)$ 
  Scan points in y-order and compare distance between
  each point and next 11 neighbors. If any of these
  distances is less than  $\delta$ , update  $\delta$ .  $O(n)$ 
  return  $\delta$ .
}

```

3

Closest Pair of Points: Analysis

Running time.

$$T(n) \leq 2T(n/2) + O(n \log n) \Rightarrow T(n) = O(n \log^2 n)$$

Q. Can we achieve $O(n \log n)$?

- A. Yes. Don't sort points in strip from scratch each time.
- Each recursive call returns two lists: all points sorted by y coordinate, and all points sorted by x coordinate.
 - Sort by **merging** two pre-sorted lists.

$$T(n) \leq 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$$

4

5.5 Integer Multiplication

Complex Multiplication

Complex multiplication. $(a + bi)(c + di) = x + yi$.

Grade-school. $x = ac - bd$, $y = bc + ad$.

4 multiplications, 2 additions

Q. Is it possible to do with fewer multiplications?

5

Complex Multiplication

Complex multiplication. $(a + bi)(c + di) = x + yi$.

Grade-school. $x = ac - bd, y = bc + ad$.

4 multiplications, 2 additions

Q. Is it possible to do with fewer multiplications?

A. Yes. [Gauss] $x = ac - bd, y = (a + b)(c + d) - ac - bd$.

3 multiplications, 5 additions

Remark. Improvement if no hardware multiply.

Integer Addition

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

Grade-school. $\Theta(n)$ bit operations.

1	1	1	1	1	1	0	1	
+	1	1	0	1	0	1	0	1
	1	0	1	1	1	1	0	1

Remark. Grade-school addition algorithm is optimal.

Integer Multiplication

Multiplication. Given two n -bit integers a and b , compute $a \times b$.

Grade-school. $\Theta(n^2)$ bit operations.

1	1	0	1	0	1	0	1										
x	0	1	1	1	1	0	1										
	1	1	0	1	0	1	0										
	0	0	0	0	0	0	0										
	1	1	0	1	0	1	0										
	1	1	0	1	0	1	0										
	1	1	0	1	0	1	0										
	1	1	0	1	0	1	0										
	1	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	0	1

Q. Is grade-school multiplication algorithm optimal?

Divide-and-Conquer Multiplication: Warmup

To multiply two n -bit integers a and b :

- Multiply four $\frac{1}{2}n$ -bit integers, recursively.
- Add and shift to obtain result.

$$a = 2^{n/2} \cdot a_1 + a_0$$

$$b = 2^{n/2} \cdot b_1 + b_0$$

$$ab = (2^{n/2} \cdot a_1 + a_0)(2^{n/2} \cdot b_1 + b_0) = 2^n \cdot a_1 b_1 + 2^{n/2} \cdot (a_1 b_0 + a_0 b_1) + a_0 b_0$$

Ex. $a = \underbrace{10001101}_{a_1} \underbrace{}_{a_0}$ $b = \underbrace{11100001}_{b_1} \underbrace{}_{b_0}$

$$T(n) = 4T(n/2) + \Theta(n) \Rightarrow T(n) = \Theta(n^2)$$

(recursive calls) (add, shift)

Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n=0 \\ 4T(n/2) + n & \text{otherwise} \end{cases}$$

$$T(n) = \sum_{k=0}^{\lg n} n 2^k = n \left(\frac{2^{1+\lg n} - 1}{2 - 1} \right) = 2n^2 - n$$

(geometric series $\sum_{k=0}^n r^k = a \frac{1-r^{n+1}}{1-r}$)

The diagram shows a binary tree where each node $T(n)$ has four children $T(n/2)$. Each $T(n/2)$ has four children $T(n/4)$, and so on, until the base case $T(2)$. The number of nodes at each level is 4^k , and the size of each node is $n/2^k$. The total number of nodes is $4^{\lg n} = 2^{2 \lg n} = n^2$.

Karatsuba Multiplication

To multiply two n -bit integers a and b :

- Add two $\frac{1}{2}n$ bit integers.
- Multiply **three** $\frac{1}{2}n$ -bit integers, recursively.
- Add, subtract, and shift to obtain result.

$$a = 2^{n/2} \cdot a_1 + a_0$$

$$b = 2^{n/2} \cdot b_1 + b_0$$

$$ab = 2^n \cdot a_1 b_1 + 2^{n/2} \cdot (a_1 b_0 + a_0 b_1) + a_0 b_0$$

$$= 2^n \cdot a_1 b_1 + 2^{n/2} \cdot ((a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0) + a_0 b_0$$

① ② ③ ④ ⑤

Karatsuba Multiplication

To multiply two n -bit integers a and b :

- Add two $\frac{1}{2}n$ bit integers.
- Multiply **three** $\frac{1}{2}n$ -bit integers, recursively.
- Add, subtract, and shift to obtain result.

$$\begin{aligned} a &= 2^{n/2} \cdot a_1 + a_0 \\ b &= 2^{n/2} \cdot b_1 + b_0 \\ ab &= 2^n \cdot a_1 b_1 + 2^{n/2} \cdot (a_1 b_0 + a_0 b_1) + a_0 b_0 \\ &= 2^n \cdot a_1 b_1 + 2^{n/2} \cdot ((a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0) + a_0 b_0 \end{aligned}$$

①
②
③
④
⑤

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

$$T(n) \leq \underbrace{T(\lfloor n/2 \rfloor) + T(\lfloor n/2 \rfloor) + T(\lfloor n/2 \rfloor)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, subtract, shift}} \Rightarrow T(n) = O(n^{1.585}) = O(n^{1.585})$$

Karatsuba: Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n=0 \\ 3T(n/2) + n & \text{otherwise} \end{cases}$$

$$T(n) = \sum_{i=0}^{\log n} n \left(\frac{1}{2}\right)^i = n \left(\frac{1 - (\frac{1}{2})^{\log n + 1}}{1 - \frac{1}{2}} \right) = 3n^{1.585} - 2n$$

Fast Integer Division Too (!)

Integer division. Given two n -bit (or less) integers s and t , compute quotient $q = s / t$ and remainder $r = s \bmod t$.

Fact. Complexity of integer division is same as integer multiplication.

To compute quotient q :

- Approximate $x = 1 / t$ using Newton's method: $x_{i+1} = 2x_i - tx_i^2$
- After $\log n$ iterations, either $q = \lfloor sx \rfloor$ or $q = \lceil sx \rceil$.

using fast multiplication

Matrix Multiplication

Dot Product

Dot product. Given two length n vectors a and b , compute $c = a \cdot b$.

Grade-school. $\Theta(n)$ arithmetic operations.

$$\begin{aligned} a &= [.70 \quad .20 \quad .10] \\ b &= [.30 \quad .40 \quad .30] \\ a \cdot b &= (.70 \times .30) + (.20 \times .40) + (.10 \times .30) = .32 \end{aligned}$$

$a \cdot b = \sum_{i=1}^n a_i b_i$

Remark. Grade-school dot product algorithm is optimal.

Matrix Multiplication

Matrix multiplication. Given two n -by- n matrices A and B , compute $C = AB$.

Grade-school. $\Theta(n^3)$ arithmetic operations.

$$\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}$$

$$\begin{bmatrix} .59 & .32 & .41 \\ .31 & .36 & .25 \\ .45 & .31 & .42 \end{bmatrix} = \begin{bmatrix} .70 & .20 & .10 \\ .30 & .60 & .10 \\ .50 & .10 & .40 \end{bmatrix} \times \begin{bmatrix} .80 & .30 & .50 \\ .10 & .40 & .10 \\ .10 & .30 & .40 \end{bmatrix}$$

Q. Is grade-school matrix multiplication algorithm optimal? How could we use divide and conquer to multiply faster?

Block Matrix Multiplication

$$\begin{array}{c} C_{11} \\ \left[\begin{array}{cccc} 152 & 158 & 164 & 170 \\ 504 & 526 & 548 & 570 \\ 856 & 894 & 932 & 970 \\ 1208 & 1262 & 1316 & 1370 \end{array} \right] = \begin{array}{c} A_{11} \quad A_{12} \\ \left[\begin{array}{ccc} 0 & 1 & 2 \\ 4 & 5 & 6 \\ 8 & 9 & 10 \\ 12 & 13 & 14 \end{array} \right] \times \begin{array}{c} B_{11} \\ \left[\begin{array}{cccc} 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 \\ 28 & 29 & 30 & 31 \end{array} \right] \end{array}
 \end{array}$$

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} = \begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 16 & 17 \\ 20 & 21 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \times \begin{bmatrix} 24 & 25 \\ 28 & 29 \end{bmatrix} = \begin{bmatrix} 152 & 158 \\ 504 & 526 \end{bmatrix}$$

29