Divide and Conquer — Multiplying Large Integers

School Method

$$\begin{array}{rrrrrrrr}
 & & & & & 9 & 8 & 1 \\
\times & & & & 1 & 2 & 3 & 4 \\
\hline
 & & & & 3 & 9 & 2 & 4 \\
 & & & 2 & 9 & 4 & 3 & \\
 & & 1 & 9 & 6 & 2 & & \\
 & & 9 & 8 & 1 & & & \\
\hline
 & 1 & 2 & 1 & 0 & 5 & 5 & 4
\end{array}$$

This takes $O(n^2)$ time to multiply two n digit numbers.

Exercise: Time to multiply an n digit number by an m digit number is $O(nm)$.

Divide and Conquer

Easiest when both numbers have same number of digits: pad 981 to 0981.

09|81 × 12|34

	shift	
09 × 12	4	108_ _ _ _
09 × 34	2	306_ _
81 × 12	2	972_ _
81 × 34	0	2754
		1210554

recurse!

e.g. 0|9 × 1|2

	shift	
0 × 1	2	0_ _
0 × 2	1	0
9 × 1	1	9
9 × 2	0	18
		108

Apply Master Method: $T(n) = aT(\frac{n}{b}) + cn^k$

$a = 4 \quad b = 2 \quad k = 1 \qquad$ compare a to b^k

$4 > 2$

case 3

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

No gain so far!

$$T(n) = 4T(n/2) + \underbrace{O(n)}$$

time for additions and the shifts

Idea: avoid one of the four multiplications:

$$\overset{w}{9} \mid \overset{x}{81} \times \overset{y}{12} \mid \overset{z}{34} = (10^2 w + x) \times (10^2 y + z)$$

$$= 10^4 wy + 10^2(wz + xy) + xz$$

We don't need wz and xy. We just need $wz + xy$.

Consider

$$(w + x) \times (y + z) = wy + (wz + xy) + xz$$

we have these

<u>Algorithm:</u>

$p = wy$

$q = xz$

$r = (w + x) \times (y + z)$

return $10^4 p + 10^2(r - p - q) + q$

Now we get:

$$T(n) = 3T(n/2) + O(n) \quad a = 3 \quad b = 2 \quad k = 1 \qquad a = 3 > b^k = 2$$

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3}) \qquad \text{Note: } \log_2 3 \approx 1.585$$

This algorithm is was discovered by Karatsuba in 1960.

Practical issues:

1. What about number of different length? E.g., a with n digits, b with m digits, $n \gg m$

$$\overbrace{382379237492379723937927942}^{a} \times \overbrace{3422}^{b}$$

   (a) Break a into $O(n/m)$ chunks of m digits each.

   (b) Multiply each chunk by b.

   (c) Add up all products, taking into account the shifts.

   Cost: $O((n/m)m^{\log_2 3})$, or $O(nm^{0.585})$

2. Which base to use? In practice: base 2^{64}
   Number is stored as an array of 64-bit integers (unsigned long):

$$a = a_0 + a_1 2^{64} + a_2(2^{64})^2 + \cdots + a_{n-1}(2^{64})^{n-1} \longrightarrow A = \boxed{a_0 \mid a_1 \mid \cdots \mid a_{n-1}}$$

3. Asymptotically faster methods for larger n.
   Schönhage & Strassen (1971): $O(n(\log n)(\log\log n))$ (used in practice)
   recent breakthrough (2019): $O(n \log n)$

Multiplying Matrices

Problem: multiply two $n \times n$ matrices (count operations $\{+, -, \times\}$ from domain of entries)

Standard method is $O(n^3)$

Divide and Conquer: divide into submatrice of size $n/2$

$$\left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right] = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$= \left[\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{array}\right]$$

$T(n) = 8T(n/2) + O(n^2) \quad a = 8 \quad b = 2 \quad k = 2 \quad a = 8 > b^k = 4$

$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^3)$

So far, no progress.

Strassen's algorithm: (1969)

- like idea for integer multiplication
- get by with 7 subproblems instead of 8 (tricky!)

$T(n) = 7T(n/2) + O(n^2) \quad a = 7 \quad b = 2 \quad k = 2 \quad a = 7 > b^k = 4$

$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7}) \quad$ Note: $\log_2 7 \approx 2.808$

Again, there are asymptotically faster methods, but they are not considered to be practical.

The Centrality of Matrix Multiplication

Suppose two $n \times n$ matrices can be multiplied using $O(n^\omega)$: $2 \le \omega \le 3$.

Many problems can be solved in time $O(n^\omega)$:

solving $Ax = b$ computing $\det A$ computing A^{-1}.

Many problems are at least as difficult as matrix multiplication.

Example: Reduction of triangular matrix inversion to matrix multiplication.

Compute the inverse of an $n \times n$ upper triangular matrix T.

Divide and Conquer: decompose T into blocks of size $n/2$.

$$T = \left[\begin{array}{c|c} T_1 & U \\ \hline & T_2 \end{array}\right] \qquad T^{-1} = \left[\begin{array}{c|c} T_1^{-1} & -T_1^{-1} U T_2^{-1} \\ \hline & T_2^{-1} \end{array}\right]$$

$T(n) = 2T(n/2) + O(n^\omega)$ $\quad a = 2 \quad b = 2 \quad k = \omega \quad a = 2 < b^k = 2^\omega \ge 4$

$T(n) \in \Theta(n^\omega)$

Example: Reduction of matrix multiplication to triangular matrix inversion.

$$\left[\begin{array}{c|c|c} I_n & A & \\ \hline & I_n & B \\ \hline & & I_n \end{array}\right]^{-1} = \left[\begin{array}{c|c|c} I_n & -A & AB \\ \hline & I_n & -B \\ \hline & & I_n \end{array}\right]$$
