Divide and Conquer — Multiplying Large Integers

School Method

\[
\begin{array}{c}
9 8 1 \\
\times \quad 1 2 3 4 \\
\hline
3 9 2 4 \\
2 9 4 3 \\
1 9 6 2 \\
9 8 1 \\
\hline
1 2 1 0 5 5 4 \\
\end{array}
\]

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.
Apply Master Method: 

\[ T(n) = aT\left(\frac{n}{k}\right) + cn^k \]

\[
\begin{align*}
a &= 4 & b &= 2 & k &= 1 & \text{compare } a \text{ to } b^k \\
T(n) \in \Theta(n^{\log_b a}) &= \Theta(n^2) \\
\end{align*}
\]

No gain so far!

Idea: avoid one of the four multiplications:

\[
\begin{align*}
\begin{array}{c|c|c}
9 & 81 & \times \\
12 & 34 & = (10^2w + x) \times (10^2y + z) \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
&= 10^4wy + 10^2(wz + xy) + xz \\
\end{align*}
\]

We don’t need \( wz \) and \( xy \). We just need \( wz + xy \).

Consider

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

Algorithm:

\[
\begin{align*}
p &= wy \\
q &= xz \\
r &= (w + x) \times (y + z) \\
\text{return } 10^4p + 10^2(r - p - q) + q
\end{align*}
\]

\[
T(n) = 4T(n/2) + O(n)
\]
Now we get:

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

\[
T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3}) \quad \text{Note: } \log_2 3 \approx 1.585
\]

This algorithm 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, \(a \gg m\)

\[
\begin{array}{c}
382379237492379723937927942 	imes 3422
\end{array}
\]

(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):

\[a = a_0 + a_12^{64} + a_2(2^{64})^2 + \cdots + a_{n-1}(2^{64})^{n-1} \rightarrow A = a_0 \mid a_1 \mid \cdots \mid a_{n-1}\]

3. Asymptotically faster methods for larger \(n\).

Schönhage & Strassen (1971): \(O(n(\log n)(\log \log n))\) (used in practice)
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 \( n/2 \)

\[
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix} =
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\
A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22}
\end{bmatrix}
\]

\[
T(n) = 8T(n/2) + O(n^2)
\]

\[
a = 8 \quad b = 2 \quad k = 2 \quad a = 8 > b^k = 4
\]

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

So far, no progress.

Strassen’s algorithm: (1969)

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

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

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

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 \( O(n^\omega) \): \( 2 \leq \omega \leq 3 \).

Many problems can be solved in time \( O(n^\omega) \):
- solving \( Ax = b \)
- computing \( \text{det} A \)
- 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 \( n/2 \).

\[
T = \begin{bmatrix} T_1 & U \\ T_2 \end{bmatrix} \quad T^{-1} = \begin{bmatrix} T_1^{-1} & -T_1^{-1}UT_2^{-1} \\ T_2^{-1} \end{bmatrix}
\]

\[
T(n) = 2T(n/2) + O(n^\omega) \quad a = 2 \quad b = 2 \quad k = \omega \quad a = 2 < b^k = 2^\omega \geq 4
\]

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

Example: Reduction of matrix multiplication to triangular matrix inversion.

\[
\begin{bmatrix} I_n & A \\ I_n & B \end{bmatrix}^{-1} = \begin{bmatrix} I_n & -A & AB \\ I_n & -B \\ I_n \end{bmatrix}
\]