MULTIPLICATION OF LARGE INTEGERS

Some applications like modern cryptography require manipulation of integers that are over 100 decimal digits long. Since such integers are too long to fit in a single word of a modern computer, they require special treatment.

In the conventional pen-and-pencil algorithm for multiplying two n-digit integers, each of the n digits of the first number is multiplied by each of the n digits of the second number for the total of n^2 digit multiplications.

The divide-and-conquer method does the above multiplication in less than n^2 digit multiplications.

Example: $23 * 14 = (2 \cdot 10^1 + 3 \cdot 10^0) * (1 \cdot 10^1 + 4 \cdot 10^0)$

$$= (2 * 1)10^2 + (2 * 4 + 3 * 1)10^1 + (3 * 4)10^0$$

$$= 2\cdot 10^2 + 11\cdot 10^1 + 12\cdot 10^0$$

$$= 3\cdot 10^2 + 2\cdot 10^1 + 2\cdot 10^0$$

$$= 322$$

The term $(2*1+3*4)$ computed as $2*4+3*1=(2+3)*(1+4)-(2*1)-(3*4)$.Here $(2*1)$and$(3*4)$arealreadycomputedused.Soonlyonemultiplicationonlywehavetodo.

For any pair of two-digit numbers $a = a_1a_0$ and $b = b_1b_0$, their product c can be computed by the formula $c = a * b = c_2 10^2 + c_1 10^1 + c_0$,

where

$c_2 = a_1 * b_1$ is the product of their first digits,

$c_0 = a_0 * b_0$ is the product of their second digits,

$c_1=(a_1+a_0)*(b_1+b_0)-(c_2+c_0)$is the product of the sum of the

a's digits and the sum of the b's digits minus the sum of c_2 andc_0.

Now we apply this trick to multiplying two n-digit integers a and b where n is a positive even number. Let us divide both numbers in the middle to take advantage of the divide-and-conquer technique.

We denote the first half of the a's digits by a_1 and the second half by a_0; for b, the notations are b_1 and b_0, respectively. In these notations, $a = a_1a_0$ implies that $a = a_1 10^{n/2} + a_0$ and $b = b_1b_0$ implies that $b = b_1 10^{n/2} + b_0$. Therefore, taking advantage of the same trick we used for two-digit numbers, we get

$$C = a * b = (a_1 10^{n/2} + a_0) * (b_1 10^{n/2} + b_0)$$

$$= (a_1 * b_1)10^n + (a_1 * b_0 + a_0 * b_1)10^{n/2} + (a_0 * b_0)$$

$$= c_2 10^n + c_1 10^{n/2} + c_0,$$

where

$c_2 = a_1 * b_1$ is the product of their first halves,

c0== $a_0 * b_0$ is the product of their second halves,

$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$

If $n/2$ is even, we can apply the same method for computing the products c_2, c_0, and c_1. Thus, if n is a power of 2, we have a recursive algorithm for computing the product of two n-digit integers. In its pure form, the recursion is stopped when n becomes 1. It can also be stopped when we deem n small enough to multiply the numbers of that size directly.

The multiplication of n-digit numbers requires three multiplications of $n/2$-digit numbers, the recurrence for the number of multiplications $M(n)$ is $M(n) = 3M(n/2)$ for $n > 1$, $M(1) = 1$. Solving it by backward substitutions for $n = 2^k$ yields

$$M(2^k) = 3M(2^{k-1})$$

$$= 3[3M(2^{k-2})]$$
$$= 3^2M(2^{k-2})$$

$$= \ldots$$

$$= 3^iM(2^{k-i})$$

$$= \ldots$$

$$= 3^kM(2^{k-k})$$

$$= 3^k.$$

(Since $k = \log_2 n$)

$M(n) =^{2} 3^{\log_2 n} = n^{\log 3} \approx n^{1.585}.$

(On the last step, we took advantage of the following property of log^{b}arithms^{b}: $a^{\log c} = c^{\log a}$.)

Let $A(n)$ be the number of digit additions and subtractions executed by the above algorithm in multiplying two n-digit decimal integers. Besides $3A(n/2)$ of these operations needed to compute the three products of $n/2$-digit numbers, the above formulas require five additions and one subtraction. Hence, we have the recurrence

$A(n) = 3 \cdot A(n/2) + cn$ for $n > 1$, $A(1) = 1$.

By using Master Theorem, we obtain $A(n) \in \Theta(n^{\log_2 3})$,

which means that the total number of additions and subtractions have the same asymptotic order of growth as the number of multiplications.

Example: For instance: a = 2345, b = 6137,

i.e., n=4. Then $C = a * b = (23*10^2+45)*(61*10^2+37)$

$C = a * b = (a_1 10^{n/2} + a_0) * (b_1 10^{n/2} + b_0)$

$= (a_1 * b_1)10^n + (a_1 * b_0 + a_0 * b_1)10^{n/2} + (a_0 * b_0)$

$= (23 * 61)10^4 + (23 * 37 + 45 * 61)10^2 + (45 * 37)$

$= 1403 \cdot 10^4 + 3596 \cdot 10^2 + 1665$

$= 14391265$

STRASSEN'S MATRIX MULTIPLICATION

The Strassen's Matrix Multiplication find the product C of two 2×2 matrices A and B

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$
$$= \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

with just seven multiplications as opposed to the eight required by the brute-force algorithm.

where

$$
\begin{aligned}
m_1 &= (a_{00} + a_{11}) * (b_{00} + b_{11}),\\
m_2 &= (a_{10} + a_{11}) * b_{00},\\
m_3 &= a_{00} * (b_{01} - b_{11}),\\
m_4 &= a_{11} * (b_{10} - b_{00}),\\
m_5 &= (a_{00} + a_{01}) * b_{11},\\
m_6 &= (a_{10} - a_{00}) * (b_{00} + b_{01}),\\
m_7 &= (a_{01} - a_{11}) * (b_{10} + b_{11}).
\end{aligned}
$$

Thus, to multiply two 2 × 2 matrices, Strassen's algorithm makes 7 multiplications and 18 additions/subtractions, whereas the brute-force algorithm requires 8 multiplications and 4 additions. These numbers should not lead us to multiplying 2 × 2 matrices by Strassen's algorithm. Its importance stems from its *asymptotic* superiority as matrix order *n* goes to infinity.

Let *A* and *B* be two $n \times n$ matrices where *n* is a power of 2. (If *n* is not a power of 2, matrices can be padded with rows and columns of zeros.) We can divide *A*, *B*, and their product *C* into four $n/2 \times n/2$ submatrices each as follows:

$$
\left[\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right] = \left[\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right] * \left[\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right]
$$

The value C_{00} can be computed either as $A_{00} * B_{00} + A_{01} * B_{10}$ or as $M_1 + M_4 - M_5 + M_7$ where M_1, M_4, M_5, and M_7 are found by Strassen's formulas, with the numbers replaced by the corresponding submatrices. The seven products of $n/2 \times n/2$ matrices are computed recursively by Strassen's matrix multiplication algorithm.

The asymptotic efficiency of Strassen's matrix multiplication algorithm

If M(n) is the number of multiplications made by Strassen's algorithm in multiplying two n×n matrices, where n is a power of 2, The recurrence relation is $M(n) = 7M(n/2)$ for $n > 1$, $M(1)=1$.

Since $n = 2^k$,

$$M(2^k) = 7M(2^{k-1})$$

$= 7[7M(2^{k-2})]$

$= 7^2 M(2^{k-2})$

$= \ldots$

$= 7^i M(2^{k-i})$

$= \ldots$

$= 7^k M(2^{k-k}) = 7^k M(2^0) = 7^k M(1) = 7^k(1)$ (Since M(1)=1)

$M(2^k) = 7^k.$

Since $k = \log_2 n,$

$M(n) = 7^{\log_2 n}$

$= n^{\log_2 7}$

$\approx \boldsymbol{n^{2.807}}$

which is smaller than n^3 required by the brute-force algorithm.

Since this savings in the number of multiplications was achieved at the expense of making extra additions, we must check the number of additions $A(n)$ made by Strassen's algorithm. To multiply two matrices of order $n>1$, the algorithm needs to multiply seven matrices of order $n/2$ and make 18 additions/subtractions of matrices of size $n/2$; when $n = 1$, no additions are made since two numbers are simply multiplied. These observations yield the following recurrence relation:

$$A(n) = 7A(n/2) + 18(n/2)^2 \text{ for } n > 1,\ A(1) = 0.$$

By closed-form solution to this recurrence and the Master Theorem, $A(n) \in \Theta(n^{\log_2 7})$. which is A better efficiency class than $\Theta(n^3)$ of the brute-force method.

Example: Multiply the following two matrices by Strassen's matrix multiplication algorithm.

$$A=\begin{bmatrix}1 & 0 & 2 & 1\\ 4 & 1 & 1 & 0\\ 0 & 1 & 3 & 0\\ 5 & 0 & 2 & 1\end{bmatrix} \qquad B=\begin{bmatrix}0 & 1 & 0 & 1\\ 2 & 1 & 0 & 4\\ 2 & 0 & 1 & 1\\ 1 & 3 & 5 & 0\end{bmatrix}$$

Answer:

$$C=\begin{bmatrix}C_{00} & C_{01}\\ C_{10} & C_{11}\end{bmatrix}=\begin{bmatrix}A_{00} & A_{01}\\ A_{10} & A_{11}\end{bmatrix}*\begin{bmatrix}B_{00} & B_{01}\\ B_{10} & B_{11}\end{bmatrix}$$

Where $\mathbf{A_{00}}=\begin{bmatrix}1 & 0\\ 4 & 1\end{bmatrix}$ $\mathbf{A_{01}}=\begin{bmatrix}2 & 1\\ 1 & 0\end{bmatrix}$ $\mathbf{A_{10}}=\begin{bmatrix}0 & 1\\ 5 & 0\end{bmatrix}$ $\mathbf{A_{11}}=\begin{bmatrix}3 & 0\\ 2 & 1\end{bmatrix}$

$\mathbf{B_{00}}=\begin{bmatrix}0 & 1\\ 2 & 1\end{bmatrix}$ $\mathbf{B_{01}}=\begin{bmatrix}0 & 1\\ 0 & 4\end{bmatrix}$ $\mathbf{B_{10}}=\begin{bmatrix}2 & 0\\ 1 & 3\end{bmatrix}$ $\mathbf{B_{11}}=\begin{bmatrix}1 & 1\\ 5 & 0\end{bmatrix}$

$$M_1 = (A_{00}+A_{11})*(B_{00}+B_{11})=\left(\begin{bmatrix}1 & 0\\ 4 & 1\end{bmatrix}+\begin{bmatrix}3 & 0\\ 2 & 1\end{bmatrix}\right)*\left(\begin{bmatrix}0 & 1\\ 2 & 1\end{bmatrix}+\begin{bmatrix}1 & 1\\ 5 & 0\end{bmatrix}\right)=\begin{bmatrix}4 & 0\\ 6 & 2\end{bmatrix}*\begin{bmatrix}1 & 2\\ 7 & 1\end{bmatrix}=\begin{bmatrix}4 & 8\\ 20 & 14\end{bmatrix}$$

Similarly apply Strassen's matrix multiplication algorithm to find the following.

$M_2=\begin{bmatrix}[illegible] & [illegible]\\ 2 & 8\end{bmatrix}$, $M_3=\begin{bmatrix}[illegible] & [illegible]\\ -9 & 4\end{bmatrix}$, $M_4=\begin{bmatrix}[illegible] & [illegible]\\ 3 & 0\end{bmatrix}$, $M_5=\begin{bmatrix}[illegible] & [illegible]\\ 10 & 5\end{bmatrix}$, $M_6=\begin{bmatrix}[illegible] & [illegible]\\ -2 & -3\end{bmatrix}$, $M_7=\begin{bmatrix}[illegible] & [illegible]\\ -9 & -4\end{bmatrix}$

$C_{00}=\begin{bmatrix}5 & 4\\ 4 & 5\end{bmatrix}$, $C_{01}=\begin{bmatrix}-7 & 3\\ 1 & 9\end{bmatrix}$, $C_{10}=\begin{bmatrix}8 & 1\\ 5 & 8\end{bmatrix}$, $C_{11}=\begin{bmatrix}3 & 7\\ 7 & 7\end{bmatrix}$

5 4 7 3

$$\mathbf{C}=\begin{bmatrix}C_{00} & C_{01}\\ C_{10} & C_{11}\end{bmatrix}=\begin{bmatrix}4 & 5 & 1 & 9\\ 8 & 1 & 3 & 7\\ 5 & 8 & 7 & 7\end{bmatrix}$$