VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Divide and Conquer

Jiří Dvorský, Ph.D.

Presentation status to date February 24, 2025

Department of Computer Science
VSB – Technical University of Ostrava

Divide and Conquer

Divide and Conquer Solution Strategy

Master Theorem

Divide and Conquer

Multiplication of Large Integers

Multiplication of Large Integers

- Multiplication of ”ordinary” integers is handled by the processor.
- What about multiplying much larger numbers, with hundreds of digits? For example, in cryptography.
- Certainly, it would be possible to implement an algorithm similar to manual multiplication.
- Its implementation requires n^2 digit multiplications, where n is the number of digits.

$$
\begin{array}{rrr}
 & 2 & 3 \\
 & 1 & 4 \\
\hline
 & 9 & 2 \\
2 & 3 & 0 \\
\hline
3 & 2 & 2
\end{array}
$$

Question to Solve

Can this be done faster? Or is this the best possible algorithm?

Multiplication of large integers – multiplication of 23 and 14

We determine the decimal expansion of numbers

$$\begin{aligned} 23 &= 2 \cdot 10^1 + 3 \cdot 10^0 \\ 14 &= 1 \cdot 10^1 + 4 \cdot 10^0 \end{aligned}$$

And multiply both expansions with each other

$$\begin{aligned} 23 \times 14 &= \left(2 \cdot 10^1 + 3 \cdot 10^0\right) \times \left(1 \cdot 10^1 + 4 \cdot 10^0\right) \\ &= (2 \times 1) \cdot 10^2 + (2 \times 4 + 3 \times 1) \cdot 10^1 + (3 \times 4) \cdot 10^0 \\ &= 322 \end{aligned}$$

For the computation, we needed 4 multiplications (denoted by $\times$), i.e., n^2 multiplications.

Multiplication of large integers – multiplication of 23 and 14 (cont.)

The middle term (tens) can also be evaluated as follows

$$2 \times 4 + 3 \times 1 = (2 + 3) \times (1 + 4) - 2 \times 1 - 3 \times 4$$

Have we not seen the expressions 2×1 and 3×4 somewhere else?

More generally, let $a = a_1 a_0$ and $b = b_1 b_0$ then

$$c = a \times b = c_2 \cdot 10^2 + c_1 \cdot 10^1 + c_0,$$

where

- $c_2 = a_1 \times b_1$ is the product of the first digits,

Multiplication of large integers – multiplication of 23 and 14 (cont.)

- $c_0 = a_0 \times b_0$ is the product of the second digits and
- $c_1 = (a_1 + a_0) \times (b_1 + b_0) - (c_2 + c_0)$ is the product of the sums of digits a and b minus the digits c_2 and c_0.

Let us have two n-digit numbers a and b, where n is an even natural number.

We will denote the first half of the digits of the number a as a_1, the second half as a_0. The notation $a = a_1 a_0$ will be understood as

$$a = a_1 a_0 = a_1 \cdot 10^{n/2} + a_0$$

A similar relationship holds for $b = b_1 b_0$.

Multiplication of large integers – divide and conquer

The product $c = a \times b$ can be written as

$$\begin{aligned} c &= \left(a_1 \cdot 10^{n/2} + a_0\right) \times \left(b_1 \cdot 10^{n/2} + b_0\right) \\ &= (a_1 \times b_1) \cdot 10^n + (a_1 \times b_0 + a_0 \times b_1) \cdot 10^{n/2} + (a_0 \times b_0) \\ &= c_2 \cdot 10^n + c_1 \cdot 10^{n/2} + c_0, \end{aligned}$$

where

- $c_2 = a_1 \times b_1$ is the product of the first halves,
- $c_0 = a_0 \times b_0$ is the product of the second halves and
- $c_1 = (a_1 + a_0) \times (b_1 + b_0) - (c_2 + c_0)$ is the product of the sums of halves of numbers a and b minus the sum of c_2 and c_0.

The numbers c_2, c_1 and c_0 are computed in the same way – recursive algorithm.

Termination of recursion: $n = 1$ or numbers a, b can be multiplied using hardware.

Multiplication of large integers – number of multiplications

- The number of multiplications necessary for computing the product of two n-digit numbers will be denoted as $M(n)$.
- Computing the product requires 3 multiplications of numbers of half the size. Multiplication of numbers for $n = 1$ requires one multiplication. Thus

$$
\begin{aligned}
M(n) &= 3M\left(\frac{n}{2}\right) \text{ for } n > 1 \\
M(1) &= 1
\end{aligned}
$$

Multiplication of large integers – number of multiplications (cont.)

- By the method of backward substitution for $n = 2^k$ we get

$$\begin{aligned} M\left(2^k\right) &= 3M\left(2^{k-1}\right) = 3\left[3M\left(2^{k-2}\right)\right] = 3^2 M\left(2^{k-2}\right) \\ &\vdots \\ &= 3^i M(2^{k-i}) \\ &\vdots \\ &= 3^k M(2^{k-k}) = 3^k \end{aligned}$$

Multiplication of large integers – number of multiplications (cont.)

- Since $k = \log_2 n$ we further get

$$M(n) = 3^{\log_2 n} = n^{log_2 3} \approx n^{1,585}$$

Remarks

1. For logarithms, the property $a^{\log_b c} = c^{\log_b a}$ holds.
2. The recursion does not necessarily have to continue until $n = 1$, it can be stopped earlier and for small n the standard algorithm can be used.

Multiplication of large integers – number of additions and subtractions

- But what about addition and subtraction? Is the lower number of multiplications offset by a higher number of additions and multiplications?
- Let us denote $A(n)$ as the number of additions and subtractions when multiplying two n-digit numbers.
- In addition to $3A\left(\frac{n}{2}\right)$ operations necessary for the recursive computation of c_2, c_1 and c_0, we need 5 additions and 1 subtraction (marked in color on slide 319), so

$$\begin{aligned} A(n) &= 3A\left(\frac{n}{2}\right) + cn \text{ for } n > 1 \\ A(1) &= 1 \end{aligned}$$

Multiplication of large integers – number of additions and subtractions (cont.)

- According to the relation (??), the **Master theorem**, we get

$$A(n) \in \Theta\left(n^{\log_2 3}\right)$$

- The total number of additions and subtractions grows asymptotically at the same rate as the number of multiplications.

Multiplication of large integers – history

- The author of the algorithm is Soviet mathematician Anatolij Alexejevič Karacuba (1937 – 2008), who presented it in 1960.
- Until then, the prevailing opinion was that the traditional algorithm is asymptotically optimal.
- So it makes sense to deal with already “resolved” problems :-)
- The question is when to use the standard algorithm and when to use the algorithm based on the divide and conquer strategy.

Divide and Conquer

Strassen's Matrix Multiplication

Strassen's Matrix Multiplication

- Is brute force matrix multiplication the best possible strategy?
- The complexity of brute force multiplication is $\Theta(n^3)$.
- An asymptotically better algorithm was introduced by Volker Strassen in 1969.
- The initial "discovery" – multiplying square matrices of order 2 can be done with 7 multiplications, unlike 8 for brute force.

Strassen's matrix multiplication of order 2

$$
\begin{aligned}
\begin{pmatrix} c_{0,0} & c_{0,1} \\ c_{1,0} & c_{1,1} \end{pmatrix} &= \begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{pmatrix} \times \begin{pmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{pmatrix} \\
&= \begin{pmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
m_1 &= (a_{0,0} + a_{1,1})(b_{0,0} + b_{1,1}) & m_5 &= (a_{0,0} + a_{0,1})b_{1,1} \\
m_2 &= (a_{1,0} + a_{1,1})b_{0,0} & m_6 &= (a_{1,0} - a_{0,0})(b_{0,0} + b_{0,1}) \\
m_3 &= a_{0,0}(b_{0,1} - b_{1,1}) & m_7 &= (a_{0,1} - a_{1,1})(b_{1,0} + b_{1,1}) \\
m_4 &= a_{1,1}(b_{1,0} - b_{0,0})
\end{aligned}
$$

Strassen's Matrix Multiplication

- Operation counts for 2×2 matrices:

	Brute Force	Strassen
Number of multiplications	8	7
Number of additions and subtractions	4	18

- Multiplying 2×2 matrices in this way is obviously nonsense. But!

Strassen's Matrix Multiplication (cont.)

- We can reformulate the relationships to convert matrix multiplication of $n \times n$ matrices into submatrices of order $\frac{n}{2} \times \frac{n}{2}$ as follows:

$$\left(\begin{array}{c|c} C_{0,0} & C_{0,1} \\ \hline C_{1,0} & C_{1,1} \end{array}\right) = \left(\begin{array}{c|c} A_{0,0} & A_{0,1} \\ \hline A_{1,0} & A_{1,1} \end{array}\right) \times \left(\begin{array}{c|c} B_{0,0} & B_{0,1} \\ \hline B_{1,0} & B_{1,1} \end{array}\right)$$

- The submatrix $C_{0,0}$ can be computed either as

$$C_{0,0} = A_{0,0} \times B_{0,0} + A_{0,1} \times B_{1,0}$$

or as

$$C_{0,0} = M_1 + M_4 - M_5 + M_7$$

- The matrices $M_1, \dots M_7$ are computed in the same recursive manner.

Strassen's matrix multiplication – complexity analysis

The number of multiplications $M(n)$ for $n \times n$ matrices is given by the recursive equation:

$$\begin{aligned} M(n) &= 7M\left(\frac{n}{2}\right) \text{ for } n > 1 \\ M(1) &= 1 \end{aligned}$$

Assuming $n = 2^k$, we obtain

$$\begin{aligned} M\left(2^k\right) &= 7M\left(2^{k-1}\right) = 7\left[7M\left(2^{k-2}\right)\right] = 7^2 M\left(2^{k-2}\right) \\ &\vdots \\ &= 7^i M\left(2^{k-i}\right) \\ &\vdots \\ &= 7^k M(2^{k-k}) = 7^k. \end{aligned}$$

Since $k = \log_2 n$ and thus

$$M(n) = 7^{\log_2 n} = n^{\log_2 7} \approx n^{2.807} < n^3$$

Strassen's matrix multiplication – complexity analysis, addition

- But does the number of additions $A(n)$ for $n \times n$ matrices not grow too quickly?
- For multiplying $n \times n$ matrices we need:
  1. to compute 7 submatrices of order $\frac{n}{2} \times \frac{n}{2}$ and
  2. to perform 18 additions/subtractions of submatrices of order $\frac{n}{2} \times \frac{n}{2}$.

  So

$$\begin{aligned} A(n) &= 7A\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2 \quad \text{for } n > 1 \\ A(1) &= 0 \end{aligned}$$

- According to the relation (**??**), **Master theorem**, we get

$$A(n) \in \Theta\left(n^{\log_2 7}\right)$$

- It follows that Strassen's matrix multiplication has an asymptotic complexity of $\Theta\left(n^{\log_2 7}\right)$, which is less than the brute force solution.

Divide and Conquer

Closest Pair Problem

Divide and Conquer

Convex hull of a set

Thanks for your attention

Bibliography