UDC 004.021

FAST ALGORITMS FOR LONG ARITHMETIC

ANDREW TITENKOV, MAXIM MATSIUSH
Polotsk State University, Belarus

In modern software industry it is often necessary to work with large numbers. These numbers are generally longer than the dimension of standard data types. Long arithmetic is usually used to solve this problem.

Long arithmetic in computing is operations on numbers, the bit of which exceeds the length of the machine word of the computer technology. These operations are not implemented in hardware. Their implementation is done programmatically using the hardware platformof working with numbers of smaller range.

Representation. In the classic form,an array of integers is used for the realization of long arithmetic, and each number is a decimal one digit from the source of a large number.

We can use the following modification of the implementation of a large number storage. Let's choose a basisβ. Then any number γcan be represented in the following form:

$$\gamma = \alpha_n * \beta^n + \alpha_{n-1} * \beta^{n-1} + \alpha_1 * \beta + \alpha_0.$$

As a matter of convenience of working with such representations of numbers, write-back is commonly used, i.e. α_0 isusually written in the first cell of the array, α_0 in the second one, etc.

Addition. When adding, in long arithmetic corresponding bits of each large number are summed successively, and if the amount exceeds the base, thenthe excess after dividing by the base remains in a given digit, and the next digit increases by 1.One can say that the addition of two large numbers in long arithmetic is performed in a column.

Subtraction. Subtraction in long arithmetic is made similarly to addition. We consistently subtract the corresponding subtrahend bit from the minuend. The only difference is that if the corresponding bit of the minuend is lower than that of the subtrahend, we add the basis to the minuend, and from the next digit we take 1.

Multiplication. There are many algorithms for multiplication in long arithmetic:

– Simple;
– Karatsubaalgorithm;
– Toomalgorithms;
– Fouriermethod.

Simple algorithm involves calculation "in a column". This method is very simple to implement, but it has low speed.

Karatsuba algorithm refers to "fast" multiplication algorithms and allows multiplying of n-digit numbers with the complexity of computing:

$$M(n) - O\left(n^{\log_2 3}\right),$$

$$\log_2 3 = 1{,}5849.$$

For multiplying two numbers in excess of the computer word length, the Karatsuba algorithm is called recursively as long as the number does not become sufficiently small so that they can be multiplied directly. Software implementation of this algorithm implies recursion that, with very large numbers, can cause a stack overflow. With small numbers the algorithm is not efficient.

The algorithm is due for rapid multiplication, which lies in calculating the intermediate results of multiplication by primitive factors. I.e. we divide the second number (in its decimal representation) intothe numbers (0 to 9), then we compute the product of the first multiplier with each primitive factor (in the worst case, we are to make 9 multiplications of the first factor by primitive factors). The final product of the first to the second factor willbe calculated by adding the cyclic at the previous stage product of the first factor with primitive factors. Note, however, that the addition of shift is necessary to make the number of bits corresponding to the number of primitive factors corresponding bits of the second number. In fact, this method is an optimized multiplication "in a column" of on-time performance, but it requires more memory (to store intermediate results). Example:

Let us assume that we need to multiply 11111 and 12323. We divide the second factor in toprimitives. These are numbers 1, 2, 3, 2, 3. This means that we need to calculate the following intermediate results:

– 11111 x 1 = 11111;
– 11111 x 2 = 22222;
– 11111 x 3 = 33333.

Then we rewritethe primitives in reverse order and perform addition of the intermediate results with a shift:

Primitives: 3, 2, 3, 2, 1.

Summation: (33333) +(22222 << 1) + (33333 << 2) + (22222 << 3) + (11111 << 4) = 33333 + 222220 + 3333300 + 22222000 + 111110000 = 136920853.

Further optimization of the algorithm is possible. The essence of optimization is to change the principle of the partition of the second factor into primitives. Primitives will be repeated sequences of numbers. Let us take the previous example and perform multiplication (11111 and 12323). We divide the second factor into primitives. This is the sequence: 1, 23, and 23. Now, we need to calculate the following intermediate results:

– 11111 x 1 = 11111;
– 11111 x 23 = 255553.

Summation: (255553) + (25553 << 2) + (11111 << 4) = 136920853.

Exponentiation. For exponentiation a number which is raised should be multiplied by itself as many times as the exponent. However, the exponent $n > 3$ already requires large computing power. To optimize exponentiation so-called "binary exponentiationis used". The essence of the method is similar to the suggested multiplication algorithm (intermediate calculations are used). The essence of the method is the following:

Provide the exponent in the form of a sum of powers of 2.

Calculate the number of intermediate squares.

Multiply them.

Let us take a detailed look atthe method. Let's assume that we need to calculate x^n.

We provide the exponent in the form of a sum of powers of 2. Each number can be written as:

$$n = \sum_{i=0}^{n} a_i + 2^i = a_0 * 2^0 + a_1 * 2^1 + a_2 * 2^2 + \ldots + a_{n-1} * 2^{n-1} + a_n * 2^n$$

where $a \in \{0;1\}$.

Next, we compute the intermediate squares, i.e. we find x^2, x^4, etc.

Finally, we multiply the intermediate squares together. It is worth noting that the multiplication is carried out only with those intermediate results ,where a_i in the expansion of the exponent is 1.

REFERENCES

1. Knuth, Donald E. The Art of Computer Programming / Donald E. Knuth. – Vol. 2. Seminumerical Algorithms. – 3rd edition. – Addison-Wesley, 1998.
2. Zuras, Dan. On Squaring and Multiplying Large Integers / Dan Zuras // ARITH-11: IEEE Symposium on Computer Arithmetic, 1993. – P. 260 – 271.

UDC 550.8.028

MULTIPATH PROPAGATION PROBLEM ANALYSIS IN DATA TRANSMISSION SYSTEMS

ANASTASIA TSYRO, VICTOR YANUSHKEVICH
Polotsk State University, Belarus

This article deals with the problem of multipath propagation and electromagnetic wave when it is incident on a layered medium.

For the selection of tangible objects on the background of the environment in practice is usually used the reflective characteristics serving tool to optimize the electrical parameters of the probing signal.

The fall of a plane electromagnetic wave on a flat boundary of two semi-infinite anisotropic media 1 and 2 is illustrated in Figure 1 for the horizontal (perpendicular) polarization when the electric field vector is perpendicular to the plane E_0 fall, and vertical (parallel) polarization when the electric field vector lies in the plane E_0 fall.