

An Algorithm for Multiplying Numbers in Binary Lunar (Dismal) Arithmetic

Svetlana Korabelshchikova, Boris Melnikov and Dang Van Vinh

Abstract. The paper discusses an algorithm for multiplying multi-digit numbers in binary lunar arithmetic. The algorithms of multiplication and addition of multi-digit numbers in lunar arithmetic have similarities with the usual algorithms of addition and multiplication of multi-digit numbers. The difference is that the bitwise addition operation involves choosing the maximum of the terms, while the bitwise multiplication operation involves choosing the minimum. These operations can be performed using the fast Fourier transform (FFT). As studies have shown, this in the binary case significantly affects the speed of the software implementation of the algorithm. There are many options for parallelizing the corresponding calculations for the FFT algorithm, and we have considered one of them.

In the lunar arithmetic, the algorithms for adding and multiplying multi-digit numbers are similar to the usual algorithms for adding and multiplying multi-digit numbers, but the bitwise operations of addition and multiplication are replaced by operations for finding the maximum and minimum, respectively. Thus, for single-digit numbers 4 and 7, we get: $4 + 7 = 7$ and $4 \cdot 7 = 4$.

Example 1. Here is an example of adding and multiplying two multi-digit numbers in the lunar arithmetic.

When adding up, we simply select the maximum in each digit, that is $4175 + 296 = 4296$. Multiply the same two numbers 4175 and 296:

That is, $4175 \cdot 6 = 4165$, $4175 \cdot 9 = 4175$, $4175 \cdot 2 = 2222$ (we select the minimum in each digit). Next, when adding up in each digit, we select the maximum. You can learn more about the features of lunar arithmetic in [1].

		4	1	7	5
	x		2	9	6
		4	1	6	5
	+	4	1	7	5
		2	2	2	2
+		2	4	4	7
		2	4	7	6
		2	4	7	6
		2	4	7	6

Now let us move on to binary lunar arithmetic. In the binary case, addition corresponds to disjunction (max), and multiplication corresponds to conjunction (min). The following figure shows the operations of addition and multiplication for the binary case:

+	0	1	·	0	1
0	0	1	0	0	0
1	1	1	1	0	1

Example 2. Here is an example of addition and multiplication of two multi-digit binary numbers. When adding up, we select the maximum in each digit, that is $1001 + 101 = 1101$. Multiply the same two binary numbers 1001 and 101:

		1	0	0	1
	x		1	0	1
		1	0	0	1
	+	0	0	0	0
		1	0	0	1
+		1	0	1	1
		1	0	1	1
		1	0	1	1
		1	0	1	1

Next, we shall establish a correspondence between the multiplication of binary numbers in lunar arithmetic and the multiplication of polynomials. Let us represent binary numbers as coefficients of a polynomial. Let us correspond the binary number 1001 with the polynomial $x^3 + 1$, and the binary number 101 with the polynomial $x^2 + 1$. Consider their multiplication

$$(x^3 + 1) \cdot (x^2 + 1) = x^5 + x^3 + x^2 + 1.$$

The result of the multiplication in Example 2 is the binary number 101101, which corresponds to the resulting polynomial $x^5 + x^3 + x^2 + 1$. This establishes a correspondence between the multiplication of numbers in binary lunar arithmetic and the multiplication of polynomials with binary coefficients. With this correspondence, the coefficients of the product of the polynomials are calculated according to the rule

$$c_k = \max\{a_i \cdot b_j \mid i + j = k\}. \tag{1}$$

The same rule applies to the multiplication of multi-digit numbers in the binary lunar arithmetic.

Let us pay attention to the following property. There is no degree of 1 and 4 in the product of polynomials, since it cannot be represented as the sum of the

degrees of the terms included in the original polynomials. Vice versa, if the degree n can be represented in at least one way as the sum of the powers of the terms included in the original polynomials, then we have the term x^n . Thus, we can conclude whether it is possible to represent a number as a sum (1 is possible, 0 is impossible).

The full version of this paper will include:

- fast Fourier transform for the lunar arithmetic;
- estimating the asymptotic complexity of the multiplication algorithm for fast Fourier transform;
- applying the lunar arithmetic to solve the unlimited knapsack problem;
- for the fast Fourier transform, comparing the calculated average values with the values obtained using another exponentiation algorithm
- the use of different options for averaging results other than the arithmetic mean;
- extracting the square root from the formal language;
- applying the lunar arithmetic for minimizing DNF;
- applying the lunar arithmetic for calculation of the Hausdorff metric on subsets of DNF;
- some test results of the computer programs for the tasks listed above;
- parallel implementation of some algorithms from this paper, as well as the corresponding test results.

References

- [1] Applegate D., LeBrun M., Sloane N. J. A. (2011), Dismal Arithmetic, <http://arxiv.org/pdf/1107.1130v2.pdf>, 33 p.

Korabelshchikova Svetlana
Odintsovo Branch of the Moscow State Institute of International Relations,
Novosportivnaya street, 3, Odintsovo,
Moscow region, 143007, Russia
e-mail: s.korabelshchikova@odin.mgimo.ru

Melnikov Boris
Shenzhen MSU-BIT University, No. 1, International University Park Road,
Dayun New Town, Longgang District
Shenzhen, 517182, China.
e-mail: bormel@mail.ru

Dang Van Vinh
Faculty of Applied Science
Ho Chi Minh City University of Technology (HCMUT), VNU-HCM,
Ho Chi Minh City, Vietnam
e-mail: dangvinh@hcmut.edu.vn