An Implementation of Large Number Arithmetic and its Application for Classification of Self-Dual Codes

Venelin Monev,
Faculty of Mathematics and Informatics,
Veliko Tarnovo University, Bulgaria

Arithmetic of large integers is used in many branches of mathematics. There are different algorithms, which work with large integers and many of them are implemented in heavy software packages (Mathematica, Maple, etc.). Some of these packages are not free and it is difficult to use even their separate functions. For our purposes, it is necessary to create an effective package which can be sufficiently adapted to the specific conditions of the objects we use. We have to calculate formulas, associated with verifying the number of all self-dual codes of given parameters. Let C be a binary linear code and $C^{\perp} = \{x \in \mathbb{F}_2^n | (x,c) = 0, \forall c \in C\}$ be its dual code where (x,c) is the Euclidean inner product. The code C is *self-dual* if $C = C^{\perp}$. Self-dual codes are intensively studied, and one of the main tasks is their classification. Two binary codes are *equivalent* if one can be obtained from the other by a permutation of coordinates. The permutation σ is an *automorphism* of the code C, if $C = \sigma(C)$ and the set of all automorphisms of C forms a group called the *automorphism group* of C and denoted by $\mathrm{Aut}(C)$. The number of codes equivalent to C is $n!/|\mathrm{Aut}(C)|$. To classify self-dual codes of length n, it is necessary to find inequivalent self-dual codes $C_1, \ldots, C_r$ so that the following mass formula holds:

$$\sum_{i=1}^{r} \frac{n!}{|\mathrm{Aut}(C_i)|} = \prod_{i=1}^{n/2-1} (2^i + 1). \tag{1}$$

There is one more mass formula for self-dual codes which uses also weights of vectors. The *weight* of a given codeword (denoted by $\mathrm{wt}(x)$) is the number of its nonzero components. The minimum weight d of a code is the smallest weight among all nonzero codewords.

Theorem 1 [4] *The family U is a complete set of representatives for equivalence classes of self-dual codes of length n and minimum weight at most d if and only if*

$$\sum_{C \in U} \frac{n!}{|\mathrm{Aut}(C)|} |\{x \in C | \mathrm{wt}(x) = d\}| = \binom{n}{d} \prod_{i=1}^{n/2-2} (2^i + 1). \tag{2}$$

A big progress has been realized recently in the classification of binary self-dual codes with different parameters. An efficient algorithm was described and then applied to length

38 in [1]. The self-dual $[40, 20, 8]$ codes were also classified [2]. Now our efforts are focused on the classification of all self-dual codes of length 40, so we need to find all codes with $d = 4$ and $d = 6$. We develop (completely) separate algorithms for $d = 4$ and $d = 6$ of the same type. One possible approach is to find codes $[40, 20, d \leq 4]$, then codes with $d = 6$, verify the calculations with mass formula and check for self-dual codes with $d = 8$. The mass formulas can be used for a verification of the obtained classification results. Algorithms for the calculation of $|Aut(C)|$ and the number of codewords with given weights are already developed [3], but the numbers that occur in the formulas overflow the machine word data types. Therefore, an additional arithmetic with large numbers is required.

The realization by C language is the most appropriate way, because we have an additional goal to use a supercomputer. Many software packages use class implementation, but in the most cases this leads to slower program execution. Therefore, simple realization by functions is preferred. We use a symbol array to store a large number. Since the sign of the result of an arithmetic operation depends of the arguments, the sign can be estimated in advance and can be stored in another variable. For convenience, one large number can be represented by structure *large* that contains an array with the digits of the number, a variable for the length and a variable for the sign.

We realized the basic arithmetic operations: addition, subtraction, multiplication and division with remainder and also a function for conversion from string to large number (and back). Addition and subtraction are based on standart arithmetic. Multiplication and division are implemented by Knuth's methods [5]. The classical methods are more efficient when the precision is less than a few thousand digits. Basing on these operations, we can easily implement the operations like squaring, modular multiplication, left and right shift with given number of positions, etc. This realization is sufficiently effective and easy for embedding in other programs. Moreover, it gets good results compared to other similar systems.

References

[1] S. Bouyuklieva and I. Bouyukliev, An algorithm for classification of binary self-dual codes, *IEEE Trans. Inform. Theory*, (to appear), arXiv:1106.5930.

[2] S. Bouyuklieva, I. Bouyukliev and M. Harada, Some extremal self-dual codes and unimodular lattices in dimension 40, submitted to *Finite Fields Appl.*, arXiv:1111.2637.

[3] I. Bouyukliev, About the code equivalence, in *Advances in Coding Theory and Cryptology*, T. Shaska, W. C. Huffman, D. Joyner, V. Ustimenko: Series on Coding Theory and Cryptology, World Scientific Publishing, Hackensack, NJ, 2007.

[4] M. Harada and A. Munemasa, Classification of self-dual codes of length 36, *Adv. Math. Commun.* **6** (2012), 229-235.

[5] D. Knuth, The Art of Computer Programming, vol.2, Addison-Wesley Professional, 1997.