# REPRESENTATION OF NUMBERS WITH NEGATIVE DIGITS AND MULTIPLICATION OF SMALL INTEGERS

**Jan-Christoph Puchta**
Mathematisches Institut, Albert-Ludwigs-Universität, Eckerstrasse 1, 79104 Freiburg, Germany
(*Submitted November 1999*)

The usual way to multiply numbers in binary representation runs as follows: To compute $m \cdot n$, copy $n$ to $x$. Multiply $x$ by two. If the last digit of $m$ is 1, then add $n$ to $x$. Now delete the last digit of $m$. Repeat until $m = 1$, then $x = m \cdot n$.

Since multiplication by 2 needs almost no time, the running time of this algorithm depends only on the time to add two numbers and the number of 1's occurring in $m$. If $n$ and $m$ are both $k$-bit numbers, one needs almost always $\frac{1}{2}k^2$-bit operations.

In [1], Dimitrov and Donevsky used the Zeckendorf representation to construct a number system in which, on average, less nonvanishing digits are needed to represent a number. Thus, using this representation, multiplication becomes about 3.2% faster. In this note we will give another number system, which gives an algorithm to multiply $n$-digit numbers in expected time $\frac{3}{8}n^2 + 2n$, i.e., for large numbers this algorithm is 25% faster than the usual one.

Note that, for very large numbers, Karatsuba and Ofmann gave an algorithms with running time $O(n^{1.585})$, and Schoenhagen and Strasser gave one with running time $O(n \log n \log\log n)$ [2]; however, the constants implied by the $O$-notations are so large that these algorithms have no meaning for most computations. Thus, faster multiplication of small numbers might speed up many computations.

We will write integers as a string consisting of 1, 0, and $-1$, and interpret a string $\alpha_k \alpha_{k-1} \ldots \alpha_0$ as $\sum \alpha_i 2^i$. Our algorithm will make use of the following simple statement.

***Proposition 1:***

***(a)*** Every integer $n$ has a unique representation as above with the following additional requirements: there are no three consecutive 1's, no two consecutive nonvanishing digits are $-1$'s, between a 1 and a $-1$ there are at least two 0's, and the first digit is 0 if and only if $n = 0$.

***(b)*** The expected number of nonvanishing digits in the representation of an $n$-bit number is $\frac{3}{8}(n+3)$.

***(c)*** The representation can be found by changing $\leq \frac{3}{4}n$-bits on average.

***Proof:*** First, we prove the uniqueness of this representation. Let $n$ be the least number such that there are two different representations $\alpha_k \ldots \alpha_0$ and $\beta_l \ldots \beta_0$. If $k > l$, then

$$\alpha_k \ldots \alpha_0 - \beta_l \ldots \beta_0 \geq \underbrace{100-100\ldots}_{k+1 \text{ digits}} - \underbrace{110110\ldots}_{\leq k \text{ digits}} = 1-1-1-1\ldots = 1.$$

Thus, $k = l$. Since, by the same computation, the leading digit of a positive digit is 1, deleting this digit together with the following 0's yields a counterexample of smaller absolute value, therefore inverting if necessary gives a smaller positive counterexample. However, we assumed $n$ to be minimal.

To construct this representation, begin with the ordinary binary representation of $n$. Now, since $2^k + \cdots + 4 + 2 + 1 = 2^{k+1} - 1$, we have

$$\underbrace{11\ldots11}_{k \text{ digits}} = \underbrace{100\ldots00}_{k+1 \text{ digits}} - 1.$$

Thus, replacing every string of $k$ consecutive 1's as above does not change the value of the string, and it is easily seen that the new representation fulfills all requirements, if we replace only blocks of length $\geq 3$. During this replacement, we have to change $k+1$ digits for every block of length $k$. Since the expected value of the number of blocks of length $k$ in an $n$-digit number is $n2^{-k-1}$, the expected value of replacements equals

$$\sum_{k=3}^{n} n(k+1)2^{-k-1} \leq \frac{n}{4} + \frac{n}{2}\sum_{k=3}^{\infty} \frac{k}{2^k} = \frac{3}{4}n.$$

In the resulting string, there is a single 1 for every substring 011 in the ordinary binary representation, two consecutive 1's for every substring 0110 and a 1 and a $-1$ for every block of length $\geq 3$. Thus, to estimate the number of nonvanishing digits, we have to count the blocks in the ordinary binary representation. At every digit, a new block begins with probability 1/2, except the first one, where this is certain. If the last digit is 0, then there are as many 1-blocks as 0-blocks, otherwise, there is one 1-block more. Thus, the expected number of 1-blocks is $\frac{n+3}{4}$. Among these, there are $\frac{n+3}{8}$ blocks of length 1; thus, the total number of nonvanishing digits equals

$$2\frac{n+3}{4} - \frac{n+3}{8} = \frac{3}{8}(n+3).$$

Hence, all of our claims are proved. □

Now, adding and subtracting integers takes the same amount of time. Thus, to multiply two $n$-digit numbers using this modified binary system, we need $\frac{3}{8}(n+3)$ additions or subtractions on average. Each addition needs $n$ bit operations, so this part of the multiplication algorithm needs $\frac{3}{8}n^2 + \frac{9}{8}n$ steps. We also have to modify one of the two numbers to be multiplied, which takes $\frac{3}{4}n$ steps. Therefore, the total running time becomes

$$\frac{3}{8}n^2 + \frac{15}{8}n < \frac{3}{8}n^2 + 2n$$

as claimed.

For $n > 15$, we have $\frac{1}{2}n^2 > \frac{3}{8}n^2 + \frac{15}{8}n$. Thus, for numbers $> 2^{15} = 32768$, multiplying by using this number system is faster than the usual algorithm.

Note that things become even better if one has to do computations with the same number several times, since than one only has to convert the integers once. It is easily seen that in this case multiplication is always at least as fast as the standard algorithm.

## REFERENCES

1. V. S. Dimitrov & B. D. Donevsky. "Faster Multiplication of Medium Large Numbers via the Zeckendorf Representation." *The Fibonacci Quarterly* **33.1** (1995):74-77.
2. D. E. Knuth. *The Art of Computer Programming.* Volume 2: *Seminumerical Algorithms.* Reading, MA: Addition-Wesley, 1969.

AMS Classification Numbers: 11A67, 68Q25

❖❖❖