Representation of numbers with negative digits and multiplication of small integers

Jan-Christoph Schlage-Puchta

November 2, 2012

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 untill $m = 1$, 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 occuring 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 in average less nonvanishing digits are needed to represent a number. Thus using this representation multplication 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 $\alpha_k\alpha_{k-1}\cdots\alpha_0$ as $\sum \alpha_i 2^i$. Our algorithm will make use of the following simple statement.

Proposition 1

1. *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* $n = 0$.
2. *The expected number of nonvanishing digits in the representation of an* n*-bit number is* $\frac{3}{8}(n+3)$.
3. *This representation can be found by changing* $\leq \frac{3}{4}n$ *bits in average.*

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

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

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, thus 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 $2^k + \ldots + 4 + 2 + 1 = 2^{k+1} - 1$, we have

$$\underbrace{11\ldots 11}_{k \text{ digits}} = \underbrace{100\ldots 00{-}1}_{k+1 \text{ digits}}$$

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 fullfills all requirements, if we replace only blocks of length ≥ 3. During this replacements, 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 ≥ 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 $2\frac{n+3}{4} - \frac{n+3}{8} = \frac{3}{8}(n+3)$. Thus all our claims are proven. □

Now adding and substracting 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 substraction in average. Each addition needs n bit operations, thus this part of the multplication algorithm need $\frac{3}{8}n^2 + \frac{9}{8}n$ steps. We also have to modify one of the two numbers to be multiplied, which takes $\frac{3}{4}n$ steps, thus 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 $> 2^{15} = 32768$ multiplieing 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 then 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 muliplication of medium large numbers via the Zeckendorf Representation*, The Fibonacci Quarterly 33.1, 74–77 (1995)

[2] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Reading, Mass., Addison–Wesley 1969

Mathematics Subject Classification: 11A67, 68Q25