MATHEMATICS OF COMPUTATION
VOLUME 46, NUMBER 174
APRIL 1986, PAGES 603–608

Effect of Improved Multiplication Efficiency on Exponentiation Algorithms Derived from Addition Chains

By D. P. McCarthy

Abstract. The interaction between the efficiency of the basic multiplication algorithm and the addition chain used to compute x^n is studied. We conclude that either repeated multiplication by x or repeated squaring should be used and the provenance of each technique is established.

1. Introduction. Algorithms to evaluate x^n where n is a positive integer and x an object for which multiplication is defined have been designed and analyzed by a number of authors, see Fateman [1] and [2], Gentleman [3], Graham [4], Heindel [5], Horowitz [6], Knuth [7] and McCarthy [8]. These algorithms have included binomial or multinomial expansions, fast Fourier transforms, evaluation homomorphisms, repeated multiplication or squaring and multiplication sequences based on addition chains for n.

Of these algorithms, those based upon the addition chain for n, including the repeated multiplication and squaring algorithms, are distinguished by their simplicity. Since, given an addition chain for n, the only operation involved in evaluating x^n is multiplication, it is natural to inquire as to what is the effect of improving the efficiency of the multiplication. A good deal is now known about multiplication algorithms (see Knuth [7], and Winograd [9]), in particular, the asymptotic behavior of different multiplication algorithms. In this paper, we propose to examine four different addition chains for n and see how they compare as we change the efficiency of our multiplication algorithms.

2. Computational Model. We call $A_n = (1 = a_1, a_2, \ldots, a_k = n)$ an addition chain of length k for n if for all $i > 1$ there exist i_1, $i_2 < i$ such that $a_i = a_{i_1} + a_{i_2}$. This chain then is used to compute x^n as $x^n = x^{a_k} = x^{a_{k_1}} \cdot x^{a_{k_2}}$ and so on for all $i = k - 1, \ldots, 2$. In general, the size of the object x^{a_i} is a function of a_i, so that the cost of computing $x^{a_{i_1}} \cdot x^{a_{i_2}}$ depends on a_{i_1} and a_{i_2}, and, of course, the efficiency of the multiplication algorithm. We introduce this efficiency as a factor α by assigning a cost of multiplying $x^{a_{i_1}}$ by $x^{a_{i_2}}$, called $C(x^{a_{i_1}} \cdot x^{a_{i_2}})$, proportional to $a_{i_1} \cdot a_{i_2}$ to the power of α. That is:

$$C(x^{a_{i_1}} \cdot x^{a_{i_2}}) = (a_{i_1} \cdot a_{i_2})^{\alpha}, \qquad \alpha \geqslant 0. \tag{2.1}$$

Received November 26, 1984; revised April 30, 1985.
1980 *Mathematics Subject Classification*. Primary 68A20.

The justification for this costing is simply that many multiplication algorithms have a complexity or efficiency well bounded by $O(n^\beta)$, where n is a measure of the size of the multiplicands and β a positive number. For example, multiplication of n-digit numbers using the traditional algorithm is $O(n^2)$, binomial expansion yields an algorithm of $O(n^{\log_2 3})$, and the fast Fourier transform yields an algorithm of $O(n \cdot \log n \log\log n)$ which, while not of the form $O(n^\beta)$, is well bounded by $O(n)$. In contrast, fixed-precision numbers are multiplied in time of $O(n^0)$, that is, in constant time. Note that all these complexity measures assume arguments of equal size $n = a_{i_1} = a_{i_2}$, so that $\alpha = \beta/2$.

Using this costing, we define a multiplicative cost CA_n associated with a particular addition chain A_n as follows:

$$CA_n = \sum_{i=2}^{k} (a_{i_1} \cdot a_{i_2})^\alpha, \qquad a_k = n. \tag{2.2}$$

3. The Algorithms. Next we look at some exponentiation algorithms based on addition chains.

(a) *Repeated Multiplication by x.*

$$x^n = \begin{pmatrix} x & n = 1 \\ x \cdot x^{n-1} & n > 1 \end{pmatrix}, \tag{3.1}$$

which is associated with the addition chain $R_n = (1, 2, \ldots, a_i, \ldots, n)$, where $a_i = i$ for $i = 1, \ldots, n$.

(b) *Binary Algorithm or Repeated Squaring.* This algorithm is based upon the binary representation of n:

$$x^n = \begin{pmatrix} x & n = 1 \\ x^{n/2} \cdot x^{n/2} & n \text{ even} \\ x \cdot x^{n-1} & n \text{ odd } \& \; n > 1 \end{pmatrix}, \tag{3.2}$$

which defines a binary addition chain $B_n = (1, 2, \ldots, a_i, \ldots, n)$, where

$$a_i = \begin{pmatrix} 1 & i = 1 \\ a_{i-1} + 1 & a_i \text{ odd} \\ a_{i-1} + a_{i-1} & a_i \text{ even} \end{pmatrix}.$$

(c) *Factor Chain.* This is based upon the fact that either n is prime, or it may be factored into at least two numbers f_1 and f_2, $n = f_1 \cdot f_2$, in which case we raise x^{f_2} to the power of f_1 as follows:

$$x^n = \begin{pmatrix} x & n = 1 \\ x \cdot x^{n-1} & n > 1 \;\&\; \text{prime} \\ (x^{f_2})^{f_1} & n = f_1 \cdot f_2, 1 < f_1 \leqslant f_2 < n \end{pmatrix}. \tag{3.3}$$

If we have the factor addition chains for f_1 and f_2 as $F_1 = (1, \ldots, a_i', \ldots, f_1)$, and $F_2 = (1, \ldots, a_i'', \ldots, f_2)$, then we may compose a chain for n as $F_n = (1, \ldots, a_i'', \ldots, f_2, \ldots, a_i' \cdot f_2, \ldots, f_1 \cdot f_2 = n)$. We note in passing that there are two such chains, since $(x^{f_2})^{f_1} = (x^{f_1})^{f_2}$.

(d) *Power Chain.* On page 402 of Knuth [7] are displayed the first eight levels of a power tree and the procedure for extending it. Once created, this tree provides a nearly optimal short addition chain to compute all the tree node values. Thus, for

each n appearing in the tree, there will be a chain $P_n = (1, \ldots, p_i, \ldots, p_k = n)$, where for all i there is a $j < i$ such that $p_i = p_{i-1} + p_j$, $i = 2, \ldots, k$, so that

$$x^{p_i} = \begin{pmatrix} x & i = 1 \\ x^{p_{i-1}} \cdot x^{p_j} & i > 1 \end{pmatrix}. \tag{3.4}$$

We are now in a position to evaluate cost functions for these four exponentiation algorithms based on Eqs. (2.1), (3.1), (3.2), (3.3) and (3.4). We define CR_n, CB_n, CF_n and CP_n as the cost of the repeated, binary, factor and power algorithms, respectively:

From (2.2) & (3.1)

$$CR_n = (n-1)^\alpha + CR_{n-1} = \sum_{i=1}^{n-1} i^\alpha.$$

From (2.2) & (3.2)

$$CB_n = \begin{pmatrix} 0 & n = 1 \\ (n/2)^{2\alpha} + CB_{n/2} & n \text{ even} \\ (n-1)^\alpha + CB_{n-1} & n \text{ odd} \end{pmatrix}.$$

From (2.2) & (3.3)

$$CF_n = \begin{pmatrix} 0 & n = 1 \\ (n-1)^\alpha + CF_{n-1} & n \text{ prime} \\ f_2^{2\alpha} \cdot CF_{f_1} + CF_{f_2} & n = f_1 \cdot f_2 \end{pmatrix}.$$

From (2.2) & (3.4)

$$CP_n = \begin{pmatrix} 0 & n = 1 \\ (p_{k-1} \cdot p_j)^\alpha + CP_{P_{k-1}} & n = p_k > 1 \end{pmatrix}.$$

4. Comparison Between the Algorithms. The cost functions CR_n, CB_n, CF_n and CP_n may be most readily compared by plotting their values on a common set of ordinates. As was noted in the presentation of the computational model, certain values of α are associated with particular multiplication techniques; we may usefully tabulate some of these:

α	MULTIPLICAND	ALGORITHM
0	Fixed-precision integers	Traditional
.5	Infinite-precision integers	Bound on FFT
$\log 3/2$	Infinite-precision integers	Binomial expansion
1	Infinite-precision integers	Traditional
2	Dense univariate polynomials with infinite-precision coefficients.	Traditional
$v = 1, 2, 3$	Dense multivariate polynomials in v variables.	Traditional

In Figure 1 below we show these four cost functions plotted against n for values of $\alpha = 0, .5, 1, 1.5$ and from them we draw the following inferences.

1. In practical terms, there is no difference between algorithms based on the binary, factor and power tree algorithms. This being so, we may take the binary algorithm, which is the easiest to compute, as representative of all short chains.

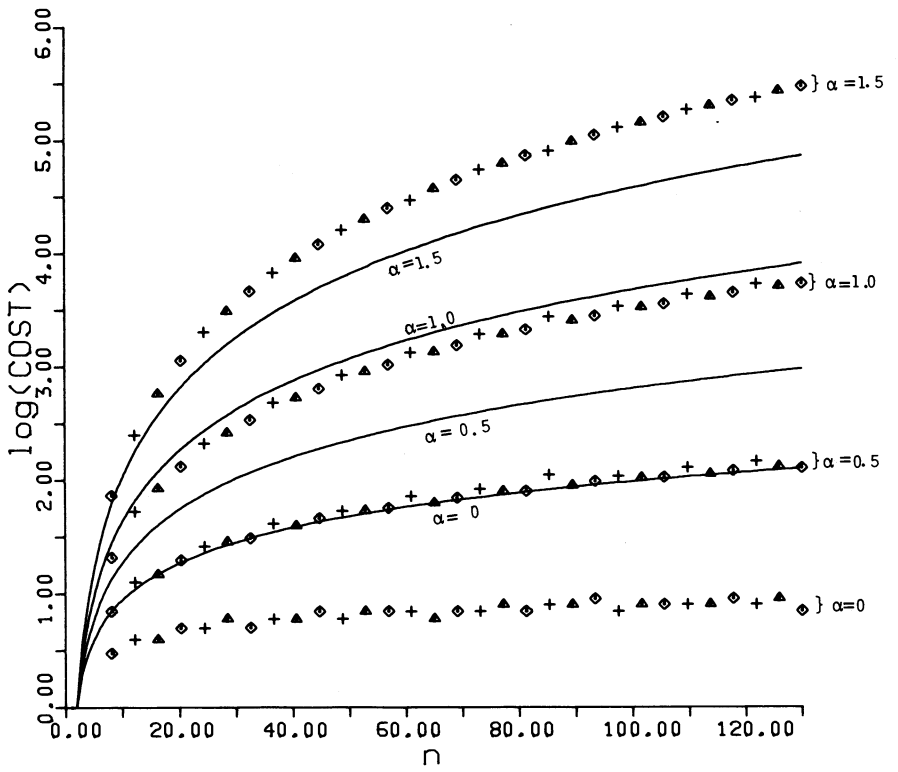

FIGURE 1
The $\log_{10}$ *of* CR = —, CB = ◇, CF = +, CP = ▵ *plotted against n for various* α.

2. For values of α less than 1 and with n increasing, repeated multiplication is an unacceptably costly method of exponentiation.

3. For $\alpha = 1$, repeated multiplication, though a little costlier, compares well with the other algorithms, particularly bearing in mind that it produces all powers of x intermediate between 1 and n.

4. For $\alpha = 1.5$, and with n increasing, repeated multiplication provides an emphatically cheaper method of evaluating x^n.

5. If we can improve our multiplication algorithm so, for example, α decreases from 1 to .5, then looking at $n = 100$ we find for $\alpha = 1$ that $CR_{100} = 5000$ and $CB_{100} = 3200$, while for $\alpha = .5$, $CR_{100} = 640$ and $CB_{100} = 100$, so that order of magnitude improvements are possible.

We now strengthen these ideas with the following theoretical results.

5. Theoretical Results. (a) $\alpha = 0$. In this case, it is clear that the shortest chain for n will yield the cheapest evaluation of x^n. Unfortunately, no simple algorithm to find even the length of this chain has emerged (see Knuth [7, pp. 402–418]). However, since the length of the binary chain at worst is less than twice the length of the shortest chain, and is generally about the same length, it represents a satisfactorily efficient algorithm for this situation.

(b) $\alpha = 1$. In Graham and Yao [4], it is shown that in this case, the binary algorithm provides absolutely the cheapest addition chain based exponentiation algorithm.

(c) $\alpha \geqslant 2$. No theoretical results have been presented for this case, but, in Gentleman [3], analysis is presented relating to the exponentiation of sparse polynomials which strongly suggests that repeated multiplication is optimal for large enough n. The following theorem strengthens this.

THEOREM. *If* $C(x^i \cdot x^j) = (i \cdot j)^\alpha$ *and* $\alpha \geqslant 2$, *then repeated multiplication provides uniquely the cheapest method by which to evaluate* x^n *for all* n.

Proof. We consider first $\alpha = 2$, where the theorem is demonstrably true for $n = 1, 2, 3$. Next, we assume it is false for $n = p + q$, where $p \geqslant q > 1$, so that we must have

$$(p \cdot q)^2 < p^2 + (p+1)^2 + \cdots + (p+q-1)^2 = \sum_{i=p}^{p+q-1} i^2,$$

or

$$(p \cdot q)^2 < p^2 q + q(q-1)p + q(q-1)(2q-1)/6,$$

implying

$$p^2 q(q-1) - q(q-1)p < q(q-1)(2q-1)/6.$$

But $q > 1$, so that $p^2 - p < (2q-1)/6$. But $p < p^2 - p$ for all $p > 2$, and $(2q-1)/6 < q$ for all q, so that we derive $p < q$ in contradiction of the assumption $p \geqslant q$. Thus, the theorem is proven true and for all $p \geqslant q > 1$ we have shown

$$(p \cdot q)^2 > \sum_{i=p}^{p+q-1} i^2. \tag{5.1}$$

We extend this result to include $\alpha > 2$ by letting $\alpha = 2 + \Delta$, $\Delta > 0$, and observing that $(p \cdot q)^\Delta > 1$ and $i^\Delta < (p \cdot q)^\Delta$ for all $i = p, \ldots, p + q - 1$. Hence, from Eq. (5.1) we have

$$(p \cdot q)^2 \cdot (p \cdot q)^\Delta > (p \cdot q)^\Delta \sum_{i=p}^{p+q-1} i^2 > \sum_{i=p}^{p+q-1} i^{2+\Delta},$$

so

$$(p \cdot q)^{2+\Delta} > \sum_{i=p}^{p+q-1} i^{2+\Delta} \quad \text{for all } \Delta > 0.$$

(d) $1 < \alpha < 2$. We examine this interval in order to observe the manner in which the advantage shifts from the binary algorithm (short chain type) to repeated multiplication as α increases. Let n_0 be the lowest integer such that repeated multiplication is cheaper or as good as the binary algorithm. That is,

$$CB_{n_0-1} < CR_{n_0-1} \quad \text{and} \quad CB_n \geqslant CR_n \quad \text{for all } n \geqslant n_0.$$

This function is best evaluated numerically and is shown below for selected values of α.

α	1.05	1.10	1.15	1.20	1.25	1.30	1.35	1.45	1.50	1.55	> 1.55
n_0	1024	206	48	22	14	10	8	6	6	1	1

It is clear from this that once $\alpha > 1.0$, repeated multiplication should be used.

6. Conclusion. This paper has compared algorithms to compute x^n derived from addition chains for n and, from the results here presented, we conclude:

1. Despite the fact there are many addition chains for most n, if we wish to minimize the multiplicative cost of evaluating x^n, we need consider only two, repeated multiplication and the binary algorithm.

2. When we characterize the cost of multiplying x^i by x^j as $(i \cdot j)^\alpha$ then, in practical terms, if $\alpha < 1$, we should use the binary algorithm and if $\alpha > 1$, we should use repeated multiplication. If $\alpha = 1$, then although repeated squaring is about twice as fast as repeated multiplication, the fact that the latter yields all the intermediate powers of x makes it the more attractive.

Department of Computer Science
Engineering School
Trinity College
Dublin 2, Ireland

1. R. J. FATEMAN, "On the computation of powers of sparse polynomials," *Stud. Appl. Math.*, v. 53, 1974, pp. 145–155.

2. R. J. FATEMAN, "Polynomial multiplication, powers and asymptotic analysis: Some comments," *SIAM J. Comput.*, v. 3, No. 3, 1974, pp. 196–213.

3. W. M. GENTLEMAN, "Optimal multiplication chains for computing a power of a symbolic polynomial," *Math. Comp.*, v. 26, 1972, pp. 945–949.

4. R. L. GRAHAM, A. C.-C. YAO & F.-F. YAO, "Addition chains with multiplicative cost," *Discrete Math.*, v. 23, no. 2, 1978, pp. 115–119.

5. L. HEINDEL, "Computation of powers of multivariate polynomials over the integers," *J. Comput. System Sci.*, v. 6, 1972, pp. 1–8.

6. E. HOROWITZ & S. SAHNI, "The computation of powers of symbolic polynomials," *SIAM J. Comput.*, v. 4, 1975, pp. 201–208.

7. D. KNUTH, *The Art of Computer Programming*—Vol. II, *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1968.

8. D. P. MCCARTHY, "The optimal algorithm to evaluate x^n using elementary multiplication methods," *Math. Comp.*, v. 31, 1977, pp. 251–256.

9. S. WINOGRAD, *Arithmetic Complexity of Computations*, CBMS–NSF Regional Conf. Series in Appl. Math., Vol. 33, SIAM, Philadelphia, PA, 1980.