MATHEMATICS OF COMPUTATION
VOLUME 46, NUMBER 173
JANUARY 1986, PAGES 255–257

Computing When Multiplications Cost Nothing

By D. J. Newman*

Abstract. A (rather strange) computer is considered which costs 1¢ to perform each addition but costs nothing to perform a multiplication. It is shown that the addition chain from 1 to n cost maximally $(\text{Log } n)^{1/2+o(1)}$ rather than the classical $\sim \text{Log } n$.

Starting with the number 1, and performing additions of numbers we already have, we find that we can reach any number we wish with exponential rapidity. Thus, we can reach the number n in exactly Log n** such addition steps when n is a power of 2, and only somewhat longer when n is not a power of 2. The obvious path to other n is via binary digits and this takes perhaps 2 Log n steps. Less obvious is Erdös' construction which gives n in $\sim$ Log n steps.

Just to be perverse, we asked how many of these addition steps were necessary if our hypothetical machine could perform multiplications absolutely free (machines are not like that but perversity is). Of course this time there is no asymptotic answer as there was before. There are very cheap numbers! Thus, every power of 2 is achieved with only 1 addition. These are the 1¢ numbers. The number 23, on the other hand is a 4¢ number, and so on. So now we must respect this difference and define $C(n)$ as the maximum cost of efficiently producing the number k, the maximum taken over all $k \leqslant n$.

Another perverse fact about this paper is that this author had virtually nothing to do with finding the answers! We did vaguely have the feeling that $C(n)$ should be "something like" $\sqrt{\text{Log } n}$, but it was Spencer*** and Rucza who separately supplied the lower and upper bounds.

Since neither of these two clever mathematicians wanted to write up the results, and since they both had such charming arguments, we decided to set them down.

THEOREM. $C(n) = (\text{Log } n)^{1/2+o(1)}$, *more precisely* $\sqrt{\text{Log } n/\text{Log Log } 2n} \leqslant C(n) \leqslant 5\sqrt{\text{Log } n}$.

Spencer: This was basically a bit of very incisive bookkeeping.

Let us form a picture of just what happens in an arbitrary procedure, call it P, with j addition steps and any number of multiplications. So denote by $a_v = a_v(P)$ the number which is produced at the vth addition step of the procedure. After this step there are a number of multiplications performed and numbers $a_1^{\lambda_1} a_2^{\lambda_2} \cdots a_v^{\lambda_v}$

Received August 30, 1984.
1980 *Mathematics Subject Classification*. Primary 05A99, 05--04.
*Supported in part by NSF 550-382-01.
**As usual, Log means $\log_2$.
***Also, and independently, done by Z. Tuza.

0025-5718/86 $1.00 + $.25 per page

are produced. Thereby the next addition gives $a_{v+1} = a_1^{\lambda_1} \cdots a_v^{\lambda_v} + a_1^{\mu_1} \cdots a_v^{\mu_v}$ and there are $2v$ description parameters (the exponents λ_i and μ_i). Finally, after all j additions have been made, there are multiplications performed and this results in $a_1^{\tau_1} a_2^{\tau_2} \cdots a_j^{\tau_j}$, and there are j more descriptive parameters.

The total number of parameters, then, is $2 + 4 + 6 + \cdots + 2(j-1) + j = j^2$, and the procedure P is exactly described by the j^2-tuple of these parameters. (For example, the 4-tuple $(0, 3, 2, 2)$ means the procedure: Form $1 + 1$, form $1 + 2^3$, form $2^2 \cdot (1 + 2^3)^2$, the end number being 324.)

We are now in a position to estimate the counting function of the integers costing at most $j¢$. Such a number might be formed by many procedures; some, perhaps of $< j$ additions, but we can, by ignoring some steps and ignoring duplications estimate this counting function by the number of j-addition-procedures. This is exactly equal to the number of j^2-tuples and since we may assume that all the parameters are bounded by $\operatorname{Log} n$ (or else the corresponding a^λ (a^μ) would exceed n). The result is that: The number of integers up to n which cost at most $j¢$ is $\leqslant (1 + \operatorname{Log} n)^{j^2}$.

Thus, if $(1 + \operatorname{Log} n)^{j^2} < n$ there is a number $\leqslant n$ which is left out of this tally, and this is to say $C(n) > j$. In other words, $j < \sqrt{\operatorname{Log} n / \operatorname{Log}\operatorname{Log} 2n}$ implies $j < C(n)$, and this proves that $C(n) \geqslant \sqrt{\operatorname{Log} n / \operatorname{Log}\operatorname{Log} 2n}$.

Ruzca: We fix on a number n, and call the largest triangular number $\leqslant \operatorname{Log} n$, $\binom{j}{2}$. For this j, we make a down payment of $j¢$ by producing $2, 3, 5, 9, \ldots, 1 + 2^{j-1}$. For this same $j¢$ then, we also produce S_j, the set of all products

$$2^m \prod_{\text{some } i<j} (1 + 2^i).$$

The following is essentially an old Putnam problem:

LEMMA. *If x_i decreases to 1 and $x_i \leqslant x_{i+1}^2$, then every number between 1 and $\prod x_i$ is equal to a subproduct, $\prod_v x_{i_v}$.*

Proof. The greedy algorithm! Given ξ we form its subproduct by taking, in turn, the largest (earliest) x_i that we can. Thus when $x_{i_1}, x_{i_2}, \ldots, x_{i_\nu}$ have been chosen with $x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_\nu} \leqslant \xi$ then we define $i_{\nu+1}$ as the first integer above i_ν for which $x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_{\nu+1}} \leqslant \xi$ (with the understanding that $i_{\nu+1} = \infty$ if $x_{i_1} \cdots x_{i_\nu} = \xi$).

It is an easy induction that, at each stage, $1 \leqslant \xi / x_{i_1} x_{i_2} \cdots x_{i_\nu} \leqslant x_{i_\nu}$. The convergence of the subproduct to ξ follows from the fact that $x_{i_\nu} \to 1$.

So let us return to n.

Write $\operatorname{Log} n = \binom{j}{2} + k + \theta$, j, k integral, $0 \leqslant k < j$, $0 \leqslant \theta < 1$. Clearly, $2^\theta < 2 < \prod(1 + 2^{-i})$ so that our lemma tells us that $2^\theta = \prod(1 + 2^{-i_\nu})$. Multiplying through by $2^{\binom{j}{2}+k}$ then gives

$$n = 2^m \prod_{i_\nu < j} (2^{i_\nu} + 1) \cdot \prod_{i_\nu \geqslant j} (1 + 2^{-i_\nu}),$$

where $m = k + \prod_{i<j;\, i \text{ not any } i_\nu} i$.

The number $N = 2^m \prod_{i_\nu < j} (2^{i_\nu} + 1)$ is in our set S_j, and furthermore, we have

$$0 \leqslant n - N = n\left(1 - \prod_{i_\nu \geqslant j} (1 + 2^{-i_\nu})^{-1}\right) \leqslant n\left(1 - \prod_{i \geqslant j} (1 + 2^{-i})^{-1}\right) < \frac{n}{2^{j-1}}.$$

If we then subtract off M, the highest power of 2 below $n - N$, we obtain $0 \leqslant n' = n - N - M < n/2^j$, and this n' corresponds to a j' which is strictly smaller than j.

We may now prove inductively that n is the sum of $2j - 1$ numbers from S_j. This is clearly true for $j = 1$, which entails $n = 1$, and if we assume that n' is the sum of $2j' - 1$ members of $S_{j'}$ (and hence of S_j), then $n = n' + N + M$ is the sum of $2j' - 1 + 2 \leqslant 2j - 1$ such, and the induction is complete.

Altogether, then, the total cost of n is bounded by $2j - 1 + j$, j being the cost of the set S_j. Since $3j - 1 \leqslant 5\sqrt{\binom{j}{2}}$ for $j \geqslant 2$, we obtain, finally, $C(n) \leqslant 5\sqrt{\operatorname{Log} n}$.

Department of Mathematics
Temple University
Philadelphia, Pennsylvania 19122