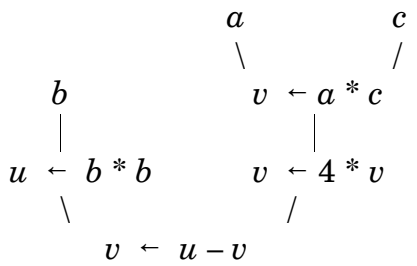
Optimality of Horner's algorithm

James T. Smith
San Francisco State University

To prove that no algorithm for evaluating polynomials requires fewer additions or multiplications than Horner's, you need a fairly precise notion of *algorithm*. The following level of detail seems sufficient. In this context, an algorithm accepts some input parameters, then performs steps of the form

$$v \leftarrow p \qquad v \leftarrow p + q \qquad v \leftarrow p * q$$

where v is a variable and p and q are operands. An operand is a variable, a constant, or a parameter. The result of the last step is the value computed by the algorithm. You can depict an algorithm as a tree. For example, if a, b, and c are parameters and u and v are variables, then you can arrange the computation $v \leftarrow b^2 - 4ac$ like this:



The leaves are labeled by parameters. A step is said to *involve* a parameter if it lies below a leaf with that label.

The algorithms considered here accept parameters $a_0, \ldots, a_n$ and x and compute the value $y = a_0 + a_1x + a_2x_2 + \cdots + a_nx^n$. Horner's algorithm uses n additions and n multiplications. The proof shows that no such algorithm can use fewer.

Before starting the proof, it's convenient to discuss some very simple facts required there. First, note that any algorithm of this type must include an addition or multiplication involving a^n: otherwise, y would be independent of that parameter.

The second preliminary fact is that any such algorithm must include a multiplication involving a_n. If it didn't, there'd be at least one addition involving a^n; consider the first. There could be no subsequent multiplications. Thus, there'd exist a positive integer k and polynomial p such that for all $a_0, \ldots, a_n$ and x,

$$y = ka_n + p(a_0, \ldots, a_{n-1}, x).$$

Replace y by its definition to get

$$a_0 + \cdots + a_n x^n = k a_n + p(a_0, \ldots, a_{n-1}, x)$$
$$a_0 + \cdots + a_{n-1} x^{n-1} - p(a_0, \ldots, a_{n-1}, x) = k a_n - a_n x^n$$

for all $a_0, \ldots, a_n$ and x. This is impossible, because the left hand side is independent of a_n, but given any x, you can choose values of a_n that change the right hand side.

The final preliminary fact is that any such algorithm must also include an addition involving a_n. If it didn't, there'd be at least one multiplication involving a_n; consider the first. There could be no subsequent additions. Thus, there'd exist a positive integer k and a polynomial p such that for all $a_0, \ldots, a_n$ and x,

$$y = a_n^k p(a_0, \ldots, a_{n-1}, x).$$

Replace y by its definition to get

$$a_0 + \cdots + a_n x^n = a_n^k p(a_0, \ldots, a_{n-1}, x)$$
$$a_0 + \cdots + a_{n-1} x^{n-1} = a_n^k p(a_0, \ldots, a_{n-1}, x) - a_n x^n.$$

for all $a_0, \ldots, a_n$ and x. This is impossible, because you can choose values of $a_0, \ldots, a_{n-1}$ and x so that the left-hand side is not zero, then set $a_n = 0$.

You can now prove the optimality result by recursion on n. It's true trivially for $n = 0$. Suppose it holds for a certain n. Let A be an algorithm that computes $a_0 + \cdots + a_{n+1} x^{n+1}$. It must include an addition involving a_{n+1}; consider the first. That must have the form $v \leftarrow a_{n+1} + w$. Construct a new algorithm A' by changing this step to $v \leftarrow w$, then replacing all remaining a_{n+1} references by references to 0. Since A' computes $a_0 + \cdots + a_n x^n$, it must include at least n additions, hence A must have included at least $n + 1$. Now consider multiplications. A must include at least one involving a_{n+1}; consider the first. That must have the form $v \leftarrow a_{n+1} * w$. Construct a new algorithm A'' by changing this step to $v \leftarrow 0$, then replacing all remaining a_{n+1} references by references to 0. Since A'' computes $a_0 + \cdots + a_n x^n$, it must include at least n multiplications, hence A must have included at least $n + 1$.

Thus Horner's algorithm is optimal, in the sense that none of the same type is faster. It's also known that any optimal algorithm performs essentially *the same* n additions and multiplications as Horner's.

For further details on this material, consult

Donald E. KNUTH, *The art of computer programming, Vol. 2: Seminumerical algorithms*. Addison-Wesley, 1969.

Lydia I. KRONSJÖ, *Algorithms: Their complexity and efficiency*. Wiley, 1979.