7-1-2007

Evaluating Polynomials

Thomas J. Harrington
University of Nebraska-Lincoln

Master of Arts in Teaching (MAT)
Masters Exam

Thomas J. Harrington

In partial fulfillment of the requirements for the Master of Arts in Teaching with a Specialization in the Teaching of Middle Level Mathematics in the Department of Mathematics.
Jim Lewis, Advisor

July 2007

Evaluating Polynomials

Thomas J. Harrington

July 2007

Evaluating Polynomials

Computers use algorithms to evaluate polynomials. This paper will study the efficiency of various algorithms for evaluating polynomials. We do this by counting the number of basic operations needed; since multiplication takes much more time to perform on a computer, we will count *only* multiplications. This paper addresses the following:

a) How many multiplications does it take to evaluate the one-variable polynomial,

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{i=0}^{n} a_i x^i$$

when the operations are performed as indicated? (Remember that powers are repeated multiplications and must be counted as such.) Write this number of multiplications as a function of *n*.

b) Use mathematical induction to prove that your answer is correct.

c) Find another way to evaluate this polynomial by doing the operations in a different order so that fewer multiplications are needed. Hint: Think of ways to intermix addition and multiplication and experiment with polynomials of lower degree. Write the number of multiplications as a new function of *n*. The best algorithm will use only *n* multiplications. Explain the algorithm you will use.

d) How many multiplications does it take to evaluate the two-variable polynomial,

$$\sum_{i=0}^{n} \sum_{j=0}^{n} a_{ij} x^i y^j$$

when the operations are performed as indicated? Write this number of multiplications as another function of *n*.

e) Use mathematical induction to prove that your answer is correct.

f) Find another way to evaluate the two-variable polynomial by doing the operations in a different order so that fewer multiplications are required. Write down the associated function of *n*. Do you think that this is the most efficient algorithm? If not hunt for a better algorithm.

Solving complex problems with has always been a time consuming process. While the invention of computers has greatly sped up the process, it has also opened the door for more complex problems. The time needed to solve complex problems with or without a computer is based on the efficiency of the algorithm. Currently one of the most time consuming mathematical problems, where an efficient algorithm does not yet exist, is the factorization of

integers, a feature of RSA public key cryptography which ensures its security (wikipedia: Integer factorization). In May of 2005 a German Federal Agency for Information Technology was able to factor an RSA-200, the RSA encryption algorithm based on a 200-digit number determined by the product of two, distinct primes. The Agency's computer took eighteen months to factor the 200-digit number into its prime factors. In computer time this is equivalent to seventy-five years of work (wikipedia: RSA-200).

This paper will explore two different algorithms for evaluating two distinct polynomials in order to find a more efficient way to evaluate them. Because the amount of time needed to compute addition does not significantly increase the time needed to evaluate a problem, only the number of multiplications will be considered.

The most basic algorithm for evaluating a polynomial is to evaluate each monomial individually and add the result. Let F(n) represent the number of multiplications needed to evaluate the polynomial: $a_0 + a_1 x + a_2 x^2 + ... + a_n x^n = \sum_{i=0}^{n} a_i x^i$ using this method.

As a first example we consider the case when n = 3. This yields the polynomial $a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3$. To count the number of multiplications required to evaluate the polynomial, we consider each term. The first term, a_0, would require no multiplications because it is a constant that will be added to the final product. The second term, a_1x^1, would require one multiplication; the third term, a_2x^2, would require two multiplications. The fourth term a_3x^3 would require three multiplications. Adding the multiplications needed to evaluate $a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3$ would be given by 1+2+3=6.

Now suppose n = 10. Then the number of multiplications needed to evaluate the polynomial $a\sum_{i=0}^{10} a_i x^i$ would be 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55.

Continuing this pattern suggests that F(n), the number of multiplications needed to evaluate a polynomial of degree n, is equal to the sum of the numbers 1 to n. The sum of the numbers 1 to n is given by the expression $\frac{n(n+1)}{2}$. Therefore F(n) = $\frac{n(n+1)}{2}$.

We prove that this formula holds true by mathematical induction:

First note that when n=1, we have the polynomial $a_0 + a_1 x^1$, so that F(1) = $\frac{1(1+1)}{2} = 1$ is true by inspection. Next we assume that F(n) = 1 + 2 + 3 + 4 + … + n = $\frac{n(n+1)}{2}$ is also true. We then need to prove that the formula holds for F(n+1); namely that 1 + 2 + … + n + (n+1) = $\frac{(n+1)(n+1+1)}{2}$.

The left hand side of this last equation can be rewritten as (1 + 2 + … + n) + (n+1). By the induction assumption (1 + 2 + … + n) is equal to $\frac{n(n+1)}{2}$. Then (1 + 2 + … + n) + (n+1) = $\frac{n(n+1)}{2}$+(n+1) = $\frac{n(n+1)+2(n+1)}{2}$. After factoring out (n+1) from the numerator we have 1+2+…+n+(n+1) = $\frac{(n+1)(n+2)}{2}$, which was what we wanted to show. Therefore by mathematical induction, F(n) = $\frac{n(n+1)}{2}$.

Next, I need to find a more efficient way to evaluate $a_0 + a_1 x + a_2 x^2 + ... + a_n x^n = \sum_{i=0}^{n} a_i x^i$.

Below we describe a more creative approach to evaluating this polynomial. Let G(n) = number of multiplications needed to evaluate the polynomial using this more efficient method. We again begin with the case when n = 3 and only consider the number of multiplications needed to evaluate this polynomial (recall that a_0 does not affect the number of multiplications). This

means I only need to count the multiplications needed to evaluate $a_1x^1+a_2x^2+a_3x^3$. Factoring out x from the polynomial creates a new polynomial of the form $x(a_1 + a_2 x^1 + a_3 x^2)$. Within the parentheses, factoring out another x from this polynomial creates a new polynomial of the form $x(a_1+x(a_2+x(a_3)))$. When the polynomial for n=3 is written in this form the inner most term has one multiplication $x*a_3$, within the second inner most parenthesis the second term has one multiplication $x*(a_2+x(a_3))$, and the final parenthesis also only has one multiplication $x*(a_1+x(a_2+x(a_3)))$. Factoring the polynomial in this fashion would only need three multiplications to evaluate the entire polynomial.

Using this approach for arbitrary n, we count the number of multiplications needed to evaluate our polynomial after factoring it in the form: $x(a_1+x(a_2+x(a_3+x(a_4+x(a_5+ \ldots+x(a_{(n-1)}+x(a_n))\ldots)$. Note that it would require n multiplications; one multiplication for every coefficient a_i, i = 1,…n. Therefore G(n) = n. This is the most efficient algorithm, since ax^n would require n multiplications and $\sum_{i=0}^{n} a_i x^i$ cannot have fewer multiplications than this.

Suppose we have another polynomial in two variables, $\sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij} x^i y^j$, and we again want to find the number of multiplications needed to evaluate this polynomial. Let P(n) equal the number of multiplications needed if we evaluate each monomial individually and add the result. For this polynomial I will again begin by counting the number of multiplications needed to evaluate each term in the case when n = 3. The polynomial would be of the form $a_{00}x^0y^0 + a_{01}x^0y^1 + a_{02}x^0y^2 + a_{03}x^0y^3 + a_{10}x^1y^0 + a_{11}x^1y^1 + a_{12}x^1y^2 + a_{13}x^1y^3 + a_{20}x^2y^0 + a_{21}x^2y^1 + a_{22}x^2y^2 + a_{23}x^2y^3 + a_{30}x^3y^0 + a_{31}x^3y^1 + a_{32}x^3y^2 + a_{33}x^3y^3$.

The number of multiplications needed to evaluate each term is shown in the following table. For example, the entry "$a_{21}x^2y^1$; 3" means that the term $a_{21}x^2y^1$ would require three multiplications in order to be evaluated.

$a_{00}x^0y^0$; 0	$a_{10}x^1y^0$; 1	$a_{20}x^2y^0$; 2	$a_{30}x^3y^0$; 3
$a_{01}x^0y^1$; 1	$a_{11}x^1y^1$; 2	$a_{21}x^2y^1$; 3	$a_{31}x^3y^1$; 4
$a_{02}x^0y^2$; 2	$a_{12}x^1y^2$; 3	$a_{22}x^2y^2$; 4	$a_{32}x^3y^2$; 5
$a_{03}x^0y^3$; 3	$a_{13}x^1y^3$; 4	$a_{23}x^2y^3$; 5	$a_{33}x^3y^3$; 6

Rewriting the table with only the number of multiplications allows us to concentrate on these values.

0	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6

In order to find the total number of multiplications for all the terms, add all the values in the table. Since 1+1+2+2+2+3+3+3+3+4+4+4+5+5+6 = 48, the number of individual multiplications needed to evaluate the polynomial is 48; i.e. P(3) = 48. Continuing the pattern for an arbitrary n yields the following table:

0	1	2	3	4	5	…	n
1	2	3	4	5	6	…	n+1
2	3	4	5	6	7	…	…
3	4	5	6	7	8	…	…

4	5	6	7	8	9	…	…
5	6	7	8	9	10	…	2n-2
…	…	…	…	…	…	2n-2	2n-1
n	n+1	…	…	…	2n-2	2n-1	2n

Therefore, summing these entries we obtain P(n) = $2(1)+3(2)+4(3)+...$

$+(n+1)(n)+(n)(n+1)+(n-1)(n+2)+\ ...+4(2n-3)+3(2n-2)+2(2n-1)+1(2n)$.

The total number of individual multiplications for P(5) can be seen in the following table for n = 0 to 5.

n	P(n)
0	0
1	4
2	18
3	48
4	100
5	180

Using these values, I created a difference table in order of find the power of the polynomial function.

n	$P(n)$	$\Delta_1 P(n)$	$\Delta_2 P(n)$	$\Delta_3 P(n)$
0	0	-	-	-
1	4	4	-	-
2	18	14	10	-

3	48	30	16	6
4	100	52	22	6
5	180	80	28	6
6	?	?	?	?

After the third difference there is a constant difference value of six. This tells me that P(n) is a cubic function. Using a calculator I entered in the data points for (n, P(n)) and ran a cubic regression. The coefficients for the standard cubic formula $y = ax^3 + bx^2 + cx + d$, were a = 1, b = 2, c = 1, and d = 0, with $R^2 = 1$. Since $R^2 = 1$ indicates a perfect correlation, we know that $P(n) = n^3 + 2n^2 + n$ is the exact formula for the number of multiplications needed to evaluate the polynomial. We test the case where n = 6. If $P(n) = n^3 + 2n^2 + n$, then $P(6) = (6)^3 + 2(6)^2 + 6$ = 216 + 72 + 6 = 294. According to the above table $\Delta_3 f_n$ = 6 so that $\Delta_2 f_6$ = 6 + 28 = 34. Then $\Delta_1 f_6$ = 34 + 80 = 114, and finally f_6 = 114 + 180 = 294. It checks. If P(n) = $n^3 + 2n^2 + n$, from here we can factor out an "n" from the polynomial and rewrite it in the form $n(n^2 + 2n \ \ +1)$ or $n(n+1)^2$. Therefore P(n) = $n(n+1)^2$.

Another option for evaluating this polynomial requires us to look back at P(n) = $2(1) + 3(2) + 4(3) + ... \ + (n+1)(n) + (n)(n+1) + (n-1)(n+2) +$ $... + 4(2n-3) + 3(2n-2) + 2(2n-1) + 1(2n)$. The first part $2(1) + 3(2) + 4(3) + ...$+(n+1)(n) can be written as $\sum_{i=1}^{n} i(i+1)$. The rest $(n)(n+1) + (n-1)(n+2) +$

$\ldots + 4(2n-3) + 3(2n-2) + 2(2n-1) + 1(2n)$ can be written as $\sum_{i=1}^{n} i(2n+1-i)$. Adding these two summations together gives us $\sum_{i=1}^{n}(i^2 + i + 2in + i - i^2) = 2(1+n)\sum_{i=1}^{n} i = 2(1+n)\frac{n(n+1)}{2} = n(n+1)^2$

We prove this by mathematical induction.

First observe that P(1) = $(1)^3 + 2(1)^2 + (1) = 1 + 2 + 1 = 4$ is true. Next assume that P(n)

$= 2(1) + 3(2) + 4(3) + \ldots + (n+1)(n) + (n)(n+1) + (n-1)(n+2) +$

$\ldots + 4(2n-3) + 3(2n-2) + 2(2n-1) + 1(2n) =$ $n(n+1)^2$ is also true (the induction hypothesis).

To assist in the final step of the induction proof, refer to the chart below:

ZONE 1 represents P(n)

ZONE 2,3,4 represents the number of multiplications added by P(n+1) above and beyond the number of multiplications counted by P(n).

ZONE 1 $n(n+1)^2$	ZONE 3 (n+1) . . . 2n+1
ZONE 2 (n+1) … 2n+1	ZONE 4 2n+2

I need to prove that P(n+1) = the number of multiplications in: ZONE 1 + ZONE 2 + ZONE 3 + ZONE 4 = $2(1) + 3(2) + 4(3) + \ldots$

$+ (n+1)(n) + (n)(n+1) + (n-1)(n+2) + \ldots + 4(2n-3) + 3(2n-2) + 2(2n-1) + 1(2n)$

+{2[(n+1)+ …+2n+1]+(2n+2)}= $(n+1)(n+1+1)^2$.

On the right hand side of the equation $(n+1)(n+1+1)^2$ = $(n+1)(n^2 + 2n + 4) =$ $n^3 + 5n^2 + 8n + 4$. On the left hand side of the equation, ZONE 1 = $n(n+1)^2$. The number of multiplications for ZONE 2 can be found by finding the sum of the numbers from 1 to (2n+1) and subtracting the sum of the numbers from 1 to n. Represented by

$\sum_{i=1}^{2n+1} i - \sum_{i=1}^{n} i = \frac{(2n+1)(2n+1+1)}{2} - \frac{n(n+1)}{2}$. ZONE 3 has the same number of multiplications of ZONE 2. Finally, ZONE 4 has only (2n + 2) multiplications. Then P(n+1) = [n(n+1)^2] +

$$[\frac{(2n+1)(2n+1+1)}{2} - \frac{n(n+1)}{2}] + [\frac{(2n+1)(2n+1+1)}{2} - \frac{n(n+1)}{2}] + [2n+2]$$

$$= n^3 + 2n^2 + n + 2[\frac{(2n+1)(2n+1+1)}{2} - \frac{n(n+1)}{2}] + 2n + 2$$

$$= n^3 + 2n^2 + 3n + 2 + [\,4n^2 + 4n + 2n + 2 - (n^2 + n)\,]$$

$$= n^3 + 2n^2 + 3n + 2 + 4n^2 + 4n + 2n + 2 - n^2 - n$$

$$= n^3 + 5n^2 + 8n + 4.$$

This was what we wanted. Therefore by mathematical induction, P(n) = $n(n+1)^2$.

Finally we seek a more efficient way to evaluate this polynomial as well. Below we describe an approach to factoring our polynomial before evaluating it. The method is similar to the one used to evaluate a one variable polynomial. Let Q(n) = Number of multiplications needed if you use this more creative approach to evaluating the polynomial. I will begin by factoring out the y values in a manner similar to the previous example [x(a_1 + x(a_2 + x(a_3 + x(a_4 + x(a_5 + …+ x($a_{(n-1)}$ + x(a_n))…)]. Let $G_n(x)$ equal the number of multiplications needed to evaluate ($a_{on}x^0 + ... + a_{nn}x^n$). Then $G_0(x)$ would equal the number of multiplications needed to evaluate ($a_{00}x^0 + ... + a_{no}x^n$) and so on. Then rewriting the polynomial in the form y($G_0(x)$ + y($G_1(x)$ + y($G_2(x)$ + … + y($G_n(x)$)…). $G_0(x)$ to $G_n(x)$ is n+1 individual polynomials that have to be evaluated since $G_0(x)$ adds one more polynomial. In addition, each G(x) has n multiplications and since there are n+1 of them, n(n+1) represents the number of multiplications needed to evaluate $G_0(x)$ to $G_n(x)$. Finally each G(x), except $G_0(x)$, is multiplied by y which adds another n multiplications to the total. Therefore Q(n) = n(n+1) + n = n^2 + 2n.

If we compare the number of operations needed to evaluating the expression using this method to that of the previous method, we see that this algorithm is much faster. Suppose we want to evaluate the polynomial when n = 10, with the first algorithm I need count every individual multiplication operation, it would have $(10)^3 + 2(10)^2 + 10 = 1210$ operations. With this new algorithm the number of multiplications decreases to $(10)^2 + 2(10) = 120$. When n = 10, this new algorithm would save the evaluator 1090 multiplication operations.

After trying different methods for factoring this polynomial and because it was found with a similar procedure for $\sum_{i=0}^{n} a_i x^i$, I feel that this is the most efficient evaluation algorithm available for $\sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij} x^i y^j$. However at this time, a proof showing that it is in fact the most efficient is unavailable.

Reference

Cohen, M., Gaughan, E. D., Knoebel, A., Kurtz, D. S., & Pengelley, D. (1991) *Student research projects in calculus.* USA: The Mathematical Association of America.

http://en.wikipedia.org/wiki/Computational_complexity

http://en.wikipedia.org/wiki/Exponential_time

http://en.wikipedia.org/wiki/Integer_factorization

http://en.wikipedia.org/wiki/Polynomial_time

http://en.wikipedia.org/wiki/RSA-200

Acknowledgments

I would like to thank Mr. David Milan from the Math Department at the University of Nebraska-Lincoln for proofing my work.