The J.C.P. Miller Recurrence for Exponentiating a Polynomial, and its q-Analog

Doron ZEILBERGER [1]

In my research on constant term conjectures, I often need to expand powers of polynomials P^m, where m is very large, and P is (usually) a polynomial of several variables. I was frustrated by the slowness of all the commercial computer algebra packages. For example, in Maple^{TM}, it takes several days to do `expand (( 1+ 3*x+ 2*x**2 )**3000)`. When I asked Herb Wilf if there was any faster way of doing it, he referred me to section 4.3 if [1] and chapter 21 of [2], but added, "It is pretty fast, but not as fast as FFT (Fast Fourier Transform)". When I looked it up, I realized that, for a *fixed* polynomial $P(x)$ of degree L, the time-complexity and space-complexity are both superior to FFT! The time-complexity of doing $P(x)^m$ using FFT is $O(mL\log_2(mL)) = O(m\log(m))$, and its space-complexity is $O(mL) = O(m)$. On the other hand, the time- and space- complexity of expanding $P(x)^m$, using the recurrence of J.C.P. Miller given below, are $O(L^2m) = O(m)$ and $O(L) = O(1)$ respectively.

Although Herb Wilf claims that [1] and [2] sold lots of copies, and nothing in them can be called *obscure*, my own inquiries among the experts of computer algebra revealed that few people are aware of the method. The main purpose of this note is to bring this beautiful and simple method to the attention of the users of computer algebra. Its second purpose is to point out a $q-$analog, that should be useful to people, like myself, who investigate $q-$constant term identities.

We all know that the coefficients of $(1+x)^m$ are given by the binomial coefficients. One way to compute them is via Pascal's triangle. Alas, this will give us lots of useless information, namely, the coefficients of $(1+x)^{m'}$ for *all* $m' < m$. A much better way is to use the *closed form* formula $\binom{m}{k} = m!/(k!(m-k)!)$ for the coefficient, let's call it $a(m,k)$, of x^k in $(1+x)^m$. An even better method is, to use the equivalent statement that $a(m,k) = \frac{m-k+1}{k}a(m,k-1)$, starting with the obvious initial condition $a(m,0) = 1$.

In the J.C.P. Miller method, one can replace $(1+x)$ by *any* polynomial $P(x)$, and get a recurrence for the coefficients $a(m,k)$ of $P(x)^m$, whose order equals the degree of $P(x)$, and for which m stays fixed, so we never have to compute values of $a(m',k')$ for $m' < m$.

Recall that $\binom{m}{k}/2^m$, the coefficient of x^k in $(\frac{1}{2}+\frac{1}{2}x)^m$ is the probability of scoring k Heads upon tossing m (fair) coins. Likewise, the probability of scoring a total of k points, upon rolling a (fair) die m times, is the coefficients of x^k in $(x+x^2+x^3+x^4+x^5+x^6)^m/6^m$. More generally, for rolling m times an $L-faced$ die, with faces marked $a_1,\ldots,a_L$, with probabilities $p_1,\ldots,p_L$, the probability of scoring a total of k points is the coefficients of x^k in $P(x)^m$, where $P(x) = p_1x^{a_1}+\ldots+p_Lx^{a_L}$.

From now, fix a polynomial $P(x) := \sum_{i=0}^{L} p_i x^i$, and let $P(x)^m := \sum_{k=0}^{mL} a(m,k)x^k$. The analog of the Pascal triangle recurrence is the following.

---

[1] Department of Mathematics, Temple University, Philadelphia, PA 19122, USA. `zeilberg@math.temple.edu` . Supported in part by the NSF. I wish to thank Herb Wilf for many useful comments.

Partial Recurrence: $a(m,k) = \sum_{i=0}^{L} p_i \, a(m-1, k-i)$.

Proof: Compare the coefficient of x^k in $P(x)^m = P(x)^{m-1}P(x)$. $\square$

The analog of $\binom{m}{k} = \frac{m-k+1}{k}\binom{m}{k-1}$ is the following L-order recurrence, that enables one to recursively compute the coefficients $a(m,k)$ of $P(x)^m$, starting with the obvious initial values $a(m,0) = p_0^m$, and $a(m,k) = 0$, for $k < 0$. It requires $2L^2m$ additions and $3L^2m$ multiplications.

The J.C.P. Miller Pure Recurrence([1][2]): Assume that $p_0 \neq 0$, then

$$a(m,k) = \frac{1}{kp_0} \sum_{i=1}^{L} p_i \left[(m+1)i - k\right] a(m, k-i) \quad .$$

Proof: The original proof in [2] was similar to the one above: compare the coefficient of x^k in $(P(x)^m)' = mP'(x)P(x)^{m-1}$. Another proof, that is easier to q-analogize, is as follows. $a(m,k)$ is the coefficient of x^0 in the Laurent polynomial $P(x)^m/x^k$. For any Laurent polynomial $f(x)$, we have that the coefficient of x^0 in $x\frac{d}{dx}f(x)$ equals 0. Hence

$$0 = C_{x^0}[x\frac{d}{dx}\frac{P(x)^{m+1}}{x^k}] = C_{x^0}\{-k[p_0+p_1x+\ldots+p_Lx^L]\frac{P(x)^m}{x^k}+(m+1)[p_1+2p_2x+\ldots Lp_Lx^{L-1}]\frac{P(x)^m}{x^{k-1}}\} =$$

$$C_{x^0}\{-k[p_0\frac{P(x)^m}{x^k}+p_1\frac{P(x)^m}{x^{k-1}}+\ldots+p_L\frac{P(x)^m}{x^{k-L}}]+(m+1)[p_1\frac{P(x)^m}{x^{k-1}}+2p_2\frac{P(x)^m}{x^{k-2}}+\ldots Lp_L\frac{P(x)^m}{x^{k-L}}]\} \quad ,$$

from which the pure recurrence follows. $\square$

Given a polynomial of several variables $P(x_1, \ldots, x_k)$, we can similarly compute the coefficient of $x_1^0 \ldots x_k^0$ in $x_1\frac{\partial}{\partial x_1}(P^{m+1}/(x^{n_1}\ldots x^{n_k}))$, to get a partial recurrence in $n_1, \ldots, n_k$, but with *fixed* m. The details are left to the readers.

Finally, here is a q-analog. The q-analog of raising a polynomial to a power, $P(x)^m$, is

$$P(x)P(qx)P(q^2x)\ldots P(q^{m-1}x) \quad .$$

The q-analog of the J.C.P. Miller Pure Recurrence: Let $P(x) = \sum_{i=0}^{L} p_i x^i$ be a polynomial, with $p_0 \neq 0$, and let

$$P(x)P(qx)\ldots P(x^{m-1}x) = \sum_{i=0}^{Lm} a(m,k)x^i \quad .$$

The coefficients $a(m,k)$ can be successively computed from the following recurrence.

$$a(m,k) = \frac{(-1)}{p_0(1-q^{-k})} \sum_{i=1}^{L} p_i \, [q^{-i} - q^{mi-k}] \, a(m, k-i) \quad .$$

Proof: Use the *Stembridge-Stanton trick*[3] that for any Laurent polynomial $F(x)$, the coefficient of x^0 in $[F(x) - F(qx)]$ equals 0, with

$$F(x) = \frac{P(q^{-1}x)P(x)\ldots P(q^{m-1}x)}{x^k} \quad ,$$

and proceed as before. □

Appendix: The following are Maple procedures, `Pxm` and `qPxm`, implementing the two recurrences. `Pxm` is much faster than both `expand(P**m)` and `taylor(P**m,x=0,L*m+1)`. The function call for `Pxm` is is `Pxm(POLY,x,m)`, where `POLY` is the polynomial $P(x)$, `x` is the variable, x, and `m` is the power. It outputs $P(x)^m$. Similarly for `qPxm`. The programs are also available via anonymous `ftp`[2] to `math.temple.edu` in directory `pub/zeilberger/programs`, by `get`ing file `Pxm.maple`. This directory contains many other of the author's Maple programs.

```
Pxm:=proc(POLY,x,m) local n,g,i,L,p,P: P:=expand(POLY/x**ldegree(POLY,x)):L:=degree(P,x):  for i from
0 to L do p[i]:=coeff(P,x,i):  od:  g:=p[0]**m:  for n from 1 to L*m do g:=g+ sum(((m+1)*'i'-n)*p['i']
* coeff(g,x,n-'i'),'i'=1..L) /n/p[0]*x**n:  od:  g:= expand( g*x**(ldegree(POLY,x)*m)):g:  end:
```

```
qPxm:=proc(POLY,x,m) local n,g,i,L,p,P: P:=expand(POLY/x**ldegree(POLY,x)):L:=degree(P,x):  for i from
0 to L do p[i]:=coeff(P,x,i):  od:  g:=p[0]**m:  for n from 1 to L*m do g:=g- normal(sum( (q**(-'i')-
q**(m*'i'-n))/(1-q**(-n))*p['i'] * coeff(g,x,n-'i'),'i'=1..L)) /p[0]*x**n:  od:  g:= expand( g*x** ( lde-
gree (POLY,x)*m)):  taylor(g, x=0, degree(g,x)+1):  end:
```

References

[1] D. E. Knuth, *"Seminumerical Algorithms"*, ("The art of Computer Programming v. **2** "), second edition, Addison Wesley, Reading, 1981.

[2] A. Nijenhuis and H. Wilf, *Combinatorial Algorithms* (2^{nd} edition), Academic Press, New York, 1978.

[3] D. Zeilberger, *A Stembridge-Stanton style proof of the Habsieger-Kadell q-Morris identity*, Discrete Math., **79**, 313-322 (1989/90).

April 18, 1994; 8 Iyar, 5754.

[2] For the sake of novices, in case they still exist, let me recall the steps. 1) `ftp math.temple.edu` 2)You logon as `anonymous`, with password [your e-mail address]. 3) You type: `cd pub/zeilberger/programs`. 4) You type: `get Pxm.maple`. (If you want all the programs, type `mget *.*`) 5) You type `quit`.