Algorithms With Exact Divisions Made Faster

Arnold Schönhage
Institut für Informatik II der Universität Bonn

May 24, 1994

[summary by François Morain]

1. Preamble

And God said:

RULE 1: *Do care about the size of @ !*
RULE 2: *Do not waste a factor of two!*
RULE 3: *Do trust the truth!*
RULE 4: *Do not raise the overall cost for speeding up rare cases – avoid lotteries!*
RULE 5: *Correctness implies termination in due time!*
RULE 6: *Don't forget the algorithms in object designs!*
RULE 7: *Clean results by approximate methods is sometimes much faster!*

and he added: *life will not be easy, since*

IRON RULE: *The development of fast algorithms is slow!*

There are several algorithms that can benefit from exact division in a ring. A typical case is Bareiss' algorithm for performing Gaussian elimination on matrices with integral entries. Another example is the computation of resultants using subresultants. All this and asymptotically fast algorithms are given in [3].

We will describe algorithms for performing exact division of elements in a ring. The setting of these algorithms is given in the recent Bible [2] (from which the Eight Commandments were taken), which contains many fast algorithms for performing arithmetic in various rings or fields. In particular, there is defined the @ symbol, which is pronounced "bounded" and is a synonym for $O(1)$.

2. Exact division

2.1. Theory. The problem is easy to state. Let f and g be two elements of a ring and suppose there exists q in the ring such that $f = qg$. We want to compute q as fast as possible. For instance, if f and g are integers, f with $2m$ words and g with m words, then classical division requires approximately $\gamma_0 m^2$ operations, where γ_0 is the typical constant measuring the cost of a basic operation in the ring.

Jebelean [1] has devised an algorithm for this that requires $\frac{1}{2}\gamma_0 m^2$ operations, by using 2-adic division starting from bottom words. In [2], a similar algorithm is given with same complexity, but starting from the higher words. It was tempting to try to unify the two approaches.

Suppose for instance that f and g are polynomials in $F[y]$, where F is some field. The algorithm of Jebelean on the lowest l coefficients has cost $\gamma_0(l+1)(l+2)/2$ and the algorithm on the upper part has cost $\gamma_0 k(k+1)/2$. Since $l = m-k$, the total cost is minimized for $l = \lfloor m/2 \rfloor$ and $k = \lceil m/2 \rceil$, with a total cost of $\frac{1}{4}\gamma_0 m^2$.

This approach can be extended to the case of integer division with suitable modifications (including a little overlap) and to $\mathbb{Z}[y_1, y_2, \ldots, y_v]$ by interpolation techniques [3].

2.2. Algorithms and examples. Write:

$$f(X) = f_{m-1}X^{m-1} + f_{m-2}X^{m-2} + \cdots + f_0,$$
$$g(X) = g_{n-1}X^{n-1} + g_{n-2}X^{n-2} + \cdots + g_0$$

so that the quotient of f by g is:

$$q(X) = q_{m-n}X^{m-n} + q_{m-n-1}X^{m-n-1} + \cdots + q_0.$$

A Maple implementation of Schönhage's algorithm for finding the coefficients q_{m-n}, $q_{m-n-1}, \ldots,$ $q_{m-n-k+1}$ is

```
# f = f[m-1]X^(m-1) + ... + f[0]
# g = g[n-1]X^(n-1) + ... + g[0]
# q = q[m-n]X^(m-n) + ... + q[0]
# 1 <= k <= m-n+1.
schtabk := proc(fx, gx, x, k)

    local q, i, j, cg, f, g, m, n;

        f:=Pol2Tab(fx, x);
        g:=Pol2Tab(gx, x);
        m:=degree(fx, x)+1;
        n:=degree(gx, x)+1;
        cg:=1/g[n-1];
        for i from m-n by -1 to m-n-k+1 do
            q[i]:=f[i+n-1]*cg;
            for j from n-2 by -1 to m-k-i do
                f[i+j]:=f[i+j]-q[i]*g[j];
            od;
        od;
        sum(q['i']*x^'i', 'i'=m-n-k+1..m-n);
end:
```

This algorithm is the ordinary algorithm for computing the quotient of two polynomials, except that we need to use the coefficients of each intermediary result `f[]` up to degree `m-n-k+1`, which saves some time. The cost of this is easily seen to be $\gamma_0 k(k+1)/2$. Note that this algorithm does not suppose that $g \mid f$, as long as k is not too large.

Jebelean's algorithm, taken from [1] is given below:

```
jebtabl := proc(fx, gx, x, l)

    local q, i, j, cg, f, g, m, n, K;

        f:=Pol2Tab(fx, x);
```

```
        g:=Pol2Tab(gx, x);
        m:=degree(fx, x)+1;
        n:=degree(gx, x)+1;
        cg:=1/g[0];
        K:=m−n+1;
        for i from 0 to l do
            q[i]:=cg*f[i];
            for j from 1 to l−i do
                f[i+j]:=f[i+j]−q[i]*g[j];
            od;
        od;
        sum(q['i']*x^'i', 'i'=0..l);
    end:
```

For the explanation of the algorithm, we refer to the original paper [1]. The cost of this procedure is $\gamma_0 l(l+1)/2$. The main procedure is:

```
    # Assumes f has degree 2m − 1, g has degree m − 1 and f = gq with q of degree m.
    ediv := proc(f, g, x)

      local m, l, k, qj, qs;

        m:=iquo(degree(f, x)+1, 2);
        l:=iquo(m, 2);
        k:=m−l;
        qj:=jebtabl(f, g, x, l);
        qs:=schtabk(f, g, x, k);
        qj+qs
    end:
```

the last missing procedure being:

```
    Pol2Tab := proc(fx, x)

      local f, i;

        for i from 0 to degree(fx, x) do
            f[i]:=coeff(fx, x, i);
        od;
        op(f)
    end:
```

Trying this, we see that:

```
    |\^/|     Maple V Release 2 (Ecole Polytechnique)
._|\|   |/|_. Copyright (c) 1981-1993 by the University of Waterloo.
 \  MAPLE  /  All rights reserved. Maple and Maple V are registered
 <____ ____>  trademarks of Waterloo Maple Software.
      |       Type ? for help.
> f:=X^7+3*X^6+6*X^5+10*X^4+10*X^3+9*X^2+7*X+4:
> g:=X^3+2*X^2+3*X+4:
> ediv(f,g,X);
```

```
                    2    3    4
1 + X + X  + X  + X
```

Bibliography

[1] Jebelean (T.). – An algorithm for exact division. *Journal of Symbolic Computation*, vol. 15, 1993, pp. 169–180.

[2] Schönhage (A.), Grotefeld (A. F.), and Vetter (E.). – *Fast algorithms – A multitape Turing machine implementation.* – BI-Wissenschaftsverlag, Mannheim, 1994.

[3] Schönhage (A.), Grotefeld (A. F.), and Vetter (E.). – A new approach to resultant computations and other algorithms with exact division. – 1994. Preprint.