An Algorithm for Division of Large Integers

ICHIRO SEMBA*

We consider a division of large integers. The ordinary pencil-and-paper method with divide-and-correct technique is well known.

This paper proposes a division algorithm determining a quotient exactly without any correction technique.

1. Introduction

We consider a division of large integers. We assume that all numbers we deal with are nonnegative. A division algorithm based on the ordinary pencil-and-paper method is discussed in Knuth[1]. The divide-and-correct technique is used in the ordinary process of long division.

When a $n+1$ digit integer y is divided by a n digit integer x, where $n>1$, $0 \leqq y/x < b$ and b is the radix of ordinary positional notation, the divide-and-correct technique makes a guess about a quotient by using the leading digit(s) of y and the leading digit of x. A good approximation to the desired answer is obtained, but is not always exact. Therefore, if necessary, the approximation has to be corrected. Let p be an approximation to a quotient $\lfloor y/x \rfloor$. If $y-px<0(\geqq x)$, then p has to be decreased(increased).

This paper proposes an algorithm determining a quotient exactly, based on the $i+1$ leading digits of y and the i leading digits of x, where $1 \leqq i \leqq n$. Computer experiments indicate that a value of 2 for i is sufficient in almost all cases, when b is large.

2. Algorithm

In this section we shall discuss an algorithm for division of a $(m+n)$-place integer by a n-place integer, giving a $(m+1)$-place quotient and a n-place remainder. The term 'n-place integer' means any integer less than b^n, where b is the radix of ordinary positional notation in which the numbers are expressed.

Example. The number 123456789 is considered to be a 9-place integer to the base 10 and also considered to be 3-place integer to the base 10^4.

We are given the following primitive operations.

(a) addition or subtraction of two-place integers, giving a two-place answer.

(b) multiplication of a two-place integer by a one-place integer, giving a two-place answer.

(c) division of a two-place integer by a two-place integer, giving a two-place quotient and a two-place remainder.

If a two-place integer is represented by a word in a computer, nearly all computers will have these three operations available. So we will construct a long-division algorithm with these primitive operations (a), (b) and (c).

We note that the answer obtained by the operation (a) or (b) is less than b^2 in our algorithm. Therefore an overflow does not result from operation (a) or (b).

First we consider the following problem.

> Let $x=(x_1x_2 \ldots x_n)_b$ and $y=(y_0y_1y_2 \ldots y_n)_b$ be nonnegative integers in radix b notation, such that $n>1$, $x_1>0$ and $y/x<b$. Find an algorithm to determine a quotient $q=\lfloor y/x \rfloor$.

Our approach is to determine a quotient q, based on the i leading digits of x and the $i+1$ leading digits of y.

Let $u_i=\lfloor (y_0y_1y_2 \ldots y_i)_b/(x_1x_2 \ldots x_i)_b \rfloor$ $(1 \leqq i \leqq n)$.

It is easily seen that this value u_i is a very good approximation to a quotient q, so long as i is reasonably large. We note that $u_n=q$.

We can compute the value u_1 by primitive operations. However it is not obvious that the values $u_i(i \geqq 2)$ can be computed by primitive operations. In the following, we show that a quotient $\lfloor y/x \rfloor$ can be computed by primitive operations.

Let $v_i=(y_0y_1y_2 \ldots y_i)_b \bmod (x_1x_2 \ldots x_i)_b$ $(1 \leqq i \leqq n)$.

The values u_1 and v_1 are computed by primitive operations. The following property shows that the values u_2 and v_2 are computed by primitive operations.

Property 2.1 Let $A=\lfloor |bv_1+y_2-u_1x_2|/(x_1x_2)_b \rfloor$ and $B=|bv_1+y_2-u_1x_2| \bmod (x_1x_2)_b$.

(1) If $bv_1+y_2-u_1x_2 \geqq 0$, then we have $u_2=u_1+A$ and $v_2=B$.

(2) If $bv_1+y_2-u_1x_2<0$, then we have $u_2=u_1-A-1$ and $v_2=(x_1x_2)_b-B$.

Proof. Since $(y_0y_1y_2)_b=u_2(x_1x_2)_b+v_2$ and $(y_0y_1)_b=u_1(x_1)_b+v_1$, it follows that

*College of General Education, Ibaraki University, 2-1-1, Bunkyo, Mito, 310 Japan.

$(y_0y_1y_2)_b = u_1(x_1x_2)_b + bv_1 + y_2 - u_1x_2.$

If $bv_1 + y_2 - u_1x_2 \geqq 0$, then we have $bv_1 + y_2 - u_1x_2 = A(x_1x_2)_b + B$.
Therefore we obtain $(y_0y_1y_2)_b = (u_1 + A)(x_1x_2)_b + B$.
This means that $u_2 = u_1 + A$ and $v_2 = B$.
If $bv_1 + y_2 - u_1x_2 < 0$, then we have $-(bv_1 + y_2 - u_1x_2) = A(x_1x_2)_b + B$.
Therefore we obtain $(y_0y_1y_2)_b = u_1(x_1x_2)_b - A(x_1x_2)_b - B = (u_1 - A - 1)(x_1x_2)_b + (x_1x_2)_b - B$.
This means that $u_2 = u_1 - A - 1$ and $v_2 = (x_1x_2)_b - B$.
This completes the proof.

The following property shows the relation u_{i-1}, v_{i-1} and u_i, $v_i (3 \leqq i \leqq n)$.

Property 2.2 Let $3 \leqq i \leqq n$.

(1) If $v_{i-1} < b$ and $bv_{i-1} + y_i - u_{i-1}x_i \geqq 0$, then we have $u_i = u_{i-1}$ and $v_i = bv_{i-1} + y_i - u_{i-1}x_i$ $(0 \leqq v_i < b^2)$.
(2) If $v_{i-1} < b$ and $bv_{i-1} + y_i - u_{i-1}x_i < 0$, then we have $u_i = u_{i-1} - 1$ and $v_i = (bv_{i-1} + y_i - u_{i-1}x_i) + (x_1x_2 \ldots x_i)_b$ $(b \leqq v_i)$.
(3) If $v_{i-1} \geqq b$, then we have $u_i = u_{i-1}$ and $v_i = bv_{i-1} + y_i - u_{i-1}x_i$ $(b \leqq v_i)$.

Proof. By the fact that

$$\begin{aligned}(y_0y_1y_2 \ldots y_i)_b &= u_i(x_1x_2 \ldots x_i)_b \\ &\quad + v_i(y_0y_1y_2 \ldots y_{i-1})_b \\ &= u_{i-1}(x_1x_2 \ldots x_{i-1})_b + v_{i-1},\end{aligned}$$

it follows that

$$\begin{aligned}(y_0y_1y_2 \ldots y_i)_b &= u_{i-1}(x_1x_2 \ldots x_i)_b \\ &\quad + bv_{i-1} + y_i - u_{i-1}x_i.\end{aligned}$$

The relation $0 \leqq v_{i-1} < b$ implies that

$$-(x_1x_2 \ldots x_i)_b < bv_{i-1} + y_i - u_{i-1}x_i < (x_1x_2 \ldots x_i)_b.$$

Therefore, if $bv_{i-1} + y_i - u_{i-1}x_i \geqq 0$, then we have

$$u_i = u_{i-1} \text{ and } v_i = bv_{i-1} + y_i - u_{i-1}x_i \ (0 \leqq v_i < b^2)$$

If $bv_{i-1} + y_i - u_{i-1}x_i < 0$, then we have

$$\begin{aligned}u_i = u_{i-1} - 1 \text{ and } v_i &= (bv_{i-1} + y_i - u_{i-1}x_i) \\ &\quad + (x_1x_2 \ldots x_i)_b \ (b \leqq v_i).\end{aligned}$$

The relation $v_{i-1} \geqq b$ implies that

$$0 \leqq bv_{i-1} + y_i - u_{i-1}x_i < (x_1x_2 \ldots x_i)b.$$

Therefore we have $u_i = u_{i-1}$ and $v_i = bv_{i-1}y_i - u_{i-1}x_i$ $(b \leqq v_i)$.
The completes the proof.

We can derive the following conclusion from Property 2.2 and the fact that $u_n = q$.

Property 2.3 Let $2 \leqq i \leqq n$.
If $v_2 < b, \ldots, v_{i-1} < b, v_i \geqq b$, then we have

$$u_i = u_{i+1} = \ldots = u_n = q.$$

Proof. Obvious.

Thus the exact quotient is obtained by primitive operations.

From Properties 2.1, 2.2 and 2.3, we can construct the following algorithm determining the quotient $q = \lfloor y/x \rfloor$.

Algorithm A. Given nonnegative integers, $x = (x_1 x_2 \ldots x_n)_b$ and $y = (y_0y_1y_2 \ldots y_n)_b$, where $n > 1$, $x_1 > 0$ and $\lfloor y/x \rfloor < b$, this algorithm computes the quotient $q = \lfloor y/x \rfloor$.

The algorithm A is written in Pascal-like notation.

```
begin
  {compute u1 and v1}
  u1 := (y0y1)b div x1; v1 := (y0y1)b mod x1;
  {compute u2 and v2}
  w := bv1 + y2 - u1x2;
  if w≧0 then begin
    u2 := u1 + w div (x1x2)b; v2 := w mod (x1x2)b;
  end else begin
    u2 := u1 - 1 - (-w) div (x1x2)b;
    v2 := (x1x2)b - (-w) mod (x1x2)b;
  end;
  i := 2;
  if u2 = 0 then goto 1;
  while (vi < b) and (i < n) do begin
    i := i + 1;
    w := bv(i-1) + yi - u(i-1)xi;
    if w < 0 then begin
      ui := u(i-1) - 1; goto 1
      end else begin
      ui := u(i-1); vi := w
    end;
  end;
  {q is determined}
1:
  q := ui;
end.
```

Now we can construct a long-division algorithm.

Algorithm B. Given nonnegative integers, $s = (s_0s_1 \ldots s_{m+n})_b$ and $t = (t_1t_2 \ldots t_n)_b$, where $m > 0$, $n > 1$, $s_0 = 0$ and $t_1 > 0$, this algorithm computes a quotient $\lfloor s/t \rfloor = (q_1q_2 \ldots q_{m+1})_b$ and a remainder s mod $t = (r_1r_2 \ldots r_n)_b$. We note that $q_1 \geqq b$ need not be considered because $s_0 = 0$ and $t_1 > 0$.

```
begin
  k := 0;
  for j := 0 to m do begin
    determine q = ⌊(sj sj+1 ... sj+n)b / (t1 t2 ... tn)b⌋
    by using algorithm A.
    k := k + 1; qk := q;
    if q > 0 then (sj ... sj+n)b := (sj ... sj+n)b
                                   - q(t1 ... tn)b;
  end;
  (r1 ... rn)b := (sm+1 ... sm+n)b;
end.
```

Property 2.4
Algorithms A and B can be implemented by using primitive operations (a), (b) and (c).
Proof. Obvious.

Example. Let $s=(0123456789)_{10}$ and $t=(1256)_{10}$.
The algorithm B is executed as follows.

j	k	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	q_1	q_2	q_3	q_4	q_5	q_6	u_1	v_1	u_2	v_2	u_3v_3
0	0	0	1	2	3	4	5	6	7	8	9											
																		1	0	1	0	0
	1											0										
		0	1	2	3	4	5	6	7	8	9											
1																						
																		12	0	10	3	9
	2											0	9									
		0	0	1	0	4	1	6	7	8	9											
2																						
																		10	0	8	8	8
	3											0	9	8								
		0	0	0	0	3	6	8	7	8	9											
3																						
																		3	0	3	0	2
	4											0	9	8	2							
		0	0	0	0	1	1	7	5	8	9											
4																						
																		11	0	9	9	9
	5											0	9	8	2	9						
		0	0	0	0	0	0	4	5	4	9											
5																						
																		4	0	3	9	3
	6											0	9	8	2	9	3					
		0	0	0	0	0	0	0	7	8	1											
6																						

3. Computer Experiments

We have generated 10000 integer pairs of x and y, where $0 \leqq y/x < b$, at random and examined the required leading digits of x to determine a quotient $\lfloor y/x \rfloor$. Let i is the number of leading didits of x. The results are obtained as follows.

b	i	1	2	3	4	5	6
10		0	8328	1494	160	17	1
100		0	9826	74	0	0	0
1000		0	9970	30	0	0	0
10000		0	9998	2	0	0	0

This means that a value of 2 for i is sufficient in almost all cases, when b is large.

Reference

1. Knuth. The Art of *Computer Programming, seminumerical algorithms, 2nd*, ed., (1981).

(Received December 25, 1985; revised September 8, 1986)