On integer division

Jean-Marc Bourguet

1 Integer division

The integer division of an integer u, the dividend, by another non nul integer v, the divisor, is the determination of two integers: the quotient q and the remainder r such that $u = qv + r$ and $|r| < |v|$. With this definition, there are two results if v is not a divisor of u. To make the result unique, one has to impose an additional condition.

Three such conditions are in use. But first a word about the notations used in this document. u/v is the real division of u by v. When an uniquifying condition is implied by the context, $u \div v$ is the quotient and u rem v is the remainder of an integer division. When the uniquifying condition is not important $u \setminus v$ is the quotient and u mod v is the remainder.

When defining the uniquifying conditions, we'll also make use the two functions:

$$\operatorname{sgn} x = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases} \qquad \operatorname{trunc} x = \begin{cases} \lceil x \rceil & x < 0 \\ 0 & x = 0 \\ \lfloor x \rfloor & x > 0 \end{cases}$$

1.1 Truncating division

The truncating division can be defined in two equivalent ways: one relating the value of q with the result of the real division (which explains the name we use), the other binding the sign of r with the sign of the dividend:

$$q = \operatorname{trunc} \frac{u}{v}$$
$$r = 0 \vee \operatorname{sgn} r = \operatorname{sgn} u$$

This way of uniquifying the result of the division brings us some properties:

$$-u \div -v = u \div v$$
$$-u \div v = u \div -v = -(u \div v)$$
$$-u \operatorname{rem} v = -(u \operatorname{rem} v)$$
$$u \operatorname{rem} -v = u \operatorname{rem} v$$

That is a lot of symmetries around the value 0; for the quotient, they are the same one that the real division has. These symmetries implies that 0 is special (for instance, there are $2v-1$ integers giving 0 when divided by v when there are only v such integers for other quotients).

1.2 Floor division

Again, two equivalent ways of defining the division:

$$q = \left\lfloor \frac{u}{v} \right\rfloor$$
$$r = 0 \vee \operatorname{sgn} r = \operatorname{sgn} v$$

The properties we get are:

$$-u \div -v = u \div v$$
$$-u \operatorname{rem} -v = -(u \operatorname{rem} v)$$
$$(u + kv) \operatorname{rem} v = u \operatorname{rem} v$$

We loose some of the symmetries around the origin, but now the remainder is periodic and 0 is now behaving like other numbers.

1.3 Positive remainder division

There is only one condition to define this division:

$$0 \leq r$$

The properties are:

$$u \div -v = -(u \div v)$$
$$u \operatorname{rem} -v = u \operatorname{rem} v$$
$$(u + kv) \operatorname{rem} v = u \operatorname{rem} v$$

Again a periodical remainder and less symmetries that for the truncating division.

To illustrate, here are some examples:

	truncating	floor	positive remainder
$7 \div 3$	2	2	2
$7 \operatorname{rem} 3$	1	1	1
$-7 \div 3$	-2	-3	-3
$-7 \operatorname{rem} 3$	-1	2	2
$7 \div -3$	-2	-3	-2
$7 \operatorname{rem} -3$	1	-2	1
$-7 \div -3$	2	2	3
$-7 \operatorname{rem} -3$	-1	-1	2

Mathematicians tend to define the version they want before using it if it is important. Programmers often don't have that luxery and have to make with what the system they use provides them. It happens that processors and languages seem to have standardized on the floor division even if the other two (they differ only for negative dividend, which seems rare) appear to

be more in use among the mathematicians. Exceptions exist, some languages even provides two remainders (the truncating and the floor one).

Going from one version to the other isn't difficult, but it can be tedious. It is just adding or substracting one to the quotient, and adding or substracting the divisor to the remainder depending on the sign of the dividend and the divisor. As an example, here is an algorithm computing the floor division assuming that the truncating one is available (in practice the most common case):

Algorithm 1 Transforming truncating division in floor division

$q \leftarrow u \div v$
$r \leftarrow u \operatorname{rem} v$
if $uv < 0$ **then** ▷ Doing $u < 0 \neq v < 0$ may be preferable as it will not overflow.
  $q \leftarrow q - 1$
  $r \leftarrow r + v$
end if

2 Long division

In this section, we'll see how to divide a $m+n$ digit number u whose digits in some base B are $(u_{m+n-1} \dots u_1 u_0)$ by a n digits number v whose digits are $(v_{n-1} \dots v_1 v_0)$ using only numbers less that B^2.

The algorithm we'll use is the *long division*, the pencil and paper method by which we have learned how to do division. We'll consider that there is an additional digit for u: u_{m+n}, currently its value is 0 and it just helps to take into account to fact that such division sometimes produces m digits and sometimes $m-1$; it will be needed in the final version of the algorithm anyway. The algorithm is thus the following:

Algorithm 2 Generic long division

for $j = m \dots 0$ **do**
  $q_j \leftarrow \left\lfloor \frac{u_{j+n}B^n + \dots + u_{j+1}B + u_j}{v_{n-1}B^{n-1} + \dots + v_1 B + v_0} \right\rfloor$
  $(u_{j+n} \dots u_{j+1} u_j) \leftarrow (u_{j+n} \dots u_{j+1} u_j) - q_j (v_{n-1} \dots v_1 v_0)$
end for

At the end, $(q_m \dots q_1 q_0)$ is the quotient and the remainder is $(u_{n-1} \dots u_1 u_0)$.

The only difficulty in applying this algorithm is getting the next digit:

$$q_j = \left\lfloor \frac{u_{j+n}B^n + \dots + u_{j+1}B + u_j}{v_{n-1}B^{n-1} + \dots + v_1 B + v_0} \right\rfloor$$

for which we known that $0 \leq q_j < B$.

Well, it is easy to do it if $n = 1$:

Algorithm 3 Long division, short divisor

for $j = m \dots 0$ **do**
  $q_j \leftarrow \left\lfloor \frac{u_{j+1}B + u_j}{v_0} \right\rfloor$
  $(u_{j+n} \dots u_{j+1} u_j) \leftarrow (u_{j+n} \dots u_{j+1} u_j) - q_j v_0$
end for

In the other cases, as we are using numbers less that B^2, the obvious idea is the use

$$\hat{q} = \left\lfloor \frac{u_n B + u_{n-1}}{v_{n-1}} \right\rfloor$$

(we are temporarily dropping the j indices). Let's try to find out how near we are from the true value. First we deduce:

$$\begin{aligned}
\hat{q} &= \left\lfloor \frac{u_n B + u_{n-1}}{v_{n-1}} \right\rfloor \\
&= \left\lfloor \frac{u_n B^n + u_{n-1} B^{n-1}}{v_{n-1} B^{n-1}} \right\rfloor \\
&= \left\lfloor \frac{u}{v_{n-1} B^{n-1}} \right\rfloor \\
&\geq \left\lfloor \frac{u}{v} \right\rfloor \\
&\geq q
\end{aligned}$$

Then bound out the other direction:

$$\begin{aligned}
\hat{q} - q &= \left\lfloor \frac{u_n B^n + u_{n-1} B^{n-1}}{v_{n-1} B^{n-1}} \right\rfloor - q \\
&= \left\lfloor \frac{u}{v_{n-1} B^{n-1}} \right\rfloor - q \\
&< \left\lfloor \frac{u}{v_{n-1} B^{n-1}} \right\rfloor - \frac{u}{v} + 1 \\
&\qquad\qquad \text{as } q = \left\lfloor \frac{u}{v} \right\rfloor > \frac{u}{v} - 1 \\
&< \frac{u}{v_{n-1} B^{n-1}} - \frac{u}{v} + 1 \\
&< \frac{u}{v} \left(\frac{v - v_{n-1} B^{n-1}}{v_{n-1} B^{n-1}} \right) + 1 \\
&< B \frac{B^{n-1}}{v_{n-1} B^{n-1}} + 1 \\
&< \frac{B}{v_{n-1}} + 1
\end{aligned}$$

so if $v_{n-1} \geq B/2$ then $0 \leq \hat{q} - q \leq 2$. Knuth [1, section 4.3.1] using

$$\hat{q} = \max(\left\lfloor \frac{u_n B^n + u_{n-1} B^{n-1}}{v_{n-1} B^{n-1}} \right\rfloor, B - 1)$$

shows that then $v_{n-1} \geq \lfloor B/2 \rfloor$ is sufficient.

In order to tighen our bound on $\hat{q}$, let's try to impose $\hat{q}(v_{n-1} B + v_{n-2}) \leq u_{j+n} B^2 + u_{j+n-1} B + u_{j+n-2}$:

$$\begin{aligned}
\hat{q}(v_{n-1} B + v_{n-2}) &\leq u_{j+n} B^2 + u_{j+n-1} B + u_{j+n-2} \\
\hat{q}(v_{n-1} B + v_{n-2}) &\leq (u_{j+n} B + u_{j+n-1}) B + u_{j+n-2} \\
\hat{q} v_{n-1} B + \hat{q} v_{n-2} &\leq (\hat{q} v_{n-1} + \hat{r}) B + u_{j+n-2} \\
\hat{q} v_{n-2} &\leq \hat{r} B + u_{j+n-2}
\end{aligned}$$

This last condition can be computed without overflow if $\hat{q} < B$ and $\hat{r} < B$ (if it isn't, the condition is false).

We'll then have

$$\begin{aligned}
\hat{q} - q &= \left\lfloor \frac{u_n B^n + u_{n-1} B^{n-1} + u_{n-2} B^{n-2}}{v_{n-1} B^{n-1} + v_{n-2} B^{n-2}} \right\rfloor - q \\
&= \left\lfloor \frac{u}{v_{n-1} B^{n-1} + v_{n-2} B^{n-2}} \right\rfloor - q \\
&< \left\lfloor \frac{u}{v_{n-1} B^{n-1} + v_{n-2} B^{n-2}} \right\rfloor - \frac{u}{v} + 1 \\
&< \frac{u}{v_{n-1} B^{n-1} + v_{n-2} B^{n-2}} - \frac{u}{v} + 1 \\
&< \frac{u}{v} \left(\frac{v - v_{n-1} B^{n-1} - v_{n-2} B^{n-2}}{v_{n-1} B^{n-1} + v_{n-2} B^{n-2}} \right) + 1 \\
&< B \frac{B^{n-2}}{v_{n-1} B^{n-1} + v_{n-2} B^{n-2}} + 1 \\
&< \frac{B}{v_{n-1} B + v_{n-2}} + 1 \\
&< 2
\end{aligned}$$

If one consider that $qv + v - 1$ can differ of $(q+1)v$ only in the last digit, one see that to be more precise, one need to take all the digits into account.

Let's modify our initial algorithm using the above properties. To take advantage of the fact that if $v_{n-1} \geq B/2$ then $0 \leq \hat{q} - q \leq 2$, we'll multiply both dividend and divisor by $\lfloor B/(v_{n-1}+1) \rfloor$ (the dividend will need the additional digit we have already introduced, the divisor won't need another digit, but its first digit will be $\geq \lfloor B/2 \rfloor$) and divide the remainder by the same value at the end. We'll first use $\hat{q} = \lfloor (u_n B + u_{n-1}) / v_{n-1} \rfloor$ then adjust $\hat{q}$ until $\hat{q} < B \wedge \hat{q} v_{n-2} \leq \hat{r} B + u_{j+n-2}$ (the condition in the algorithm is a little more complicated in order to prevent overflow). Then we are sure that $\hat{q}$ is at most one too big, so after having subtracted $\hat{q} v$ from $(u_{j+n} \dots u_{j+1} u_j)$ we check if the result had a borrow and add back v in that case. The result is algorithm 4.

For B even $\lfloor B/2 \rfloor = B/2$, the refinement loop will be executed at most twice, so unrolling the loop should be considered in an implementation.

With B odd, it is possible to get $\hat{q} = q + 3$ (for example $4878_9 = 8 \times 488_9 + 487_9$ and $\hat{q} = 48_9/4 = 12_9 = 8 + 3$). But as the additional condition we use to ensure $\hat{q} \leq q + 1$ doesn't depend on v_{n-1}, unrolling the loop twice would be enough, the add back step will handle final adjustment.

References

[1] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Addison-Wesley Longman, third edition, 1998.

[2] Henry S. Warren, Jr. *Hacker's Delight.* Addison-Wesley, 2003.

Algorithm 4 Long division, full version

1: First normalize the operand to ensure $v_{n-1} \geq \lfloor B/2 \rfloor$.
2: $d \leftarrow B \setminus (v_{n-1}+1)$
3: **if** $d > 1$ **then**
4: $(v_{n-1} \dots v_1 v_0) \leftarrow d(u_{n-1} \dots u_1 u_0)$
5: $(u_{m+n} \dots u_1 u_0) \leftarrow d(u_{m+n} \dots u_1 u_0)$
6: **end if**
7: Now determine each digits.
8: **for** $j = m \dots 0$ **do**
9: Get our estimate
10: $\hat{q} \leftarrow (u_{j+n}B + u_{j+n-1}) \setminus v_{n-1}$
11: $\hat{r} \leftarrow (u_{j+n}B + u_{j+n-1}) \bmod \hat{q}v_{n-1}$
12: Refine if needed
13: **while** $\hat{q} \geq B \vee (\hat{r} < B \wedge \hat{q}v_{n-2} > B\hat{r} + u_{j+n-2})$ **do**
14: $\hat{q} \leftarrow \hat{q} - 1$
15: $\hat{r} \leftarrow \hat{r} + v_{n-1}$
16: **end while**
17: Do the substraction
18: $b = 0$
19: **for** $i = 0 \dots n-1$ **do**
20: $b \leftarrow u_{j+i} - v_i\hat{q} + b$
21: $u_{j+i} \leftarrow b \bmod B$
22: $b \leftarrow b \setminus B$
23: **end for**
24: If there is a borrow, that mean the refinement wasn't enough to get it right, add back v and adjust $\hat{q}$.
25: **if** $b \neq 0$ **then**
26: $(u_{j+n-1} \dots u_{j+1}u_j) \leftarrow (u_{j+n-1} \dots u_{j+1}u_j) + (v_{n-1} \dots v_1 v_0)$
27: $\hat{q} \leftarrow \hat{q} - 1$
28: **end if**
29: $q_j \leftarrow \hat{q}$
30: **end for**
31: Finally undo the adjustment of the remainder
32: **if** $d > 1$ **then**
33: $(u_{n-1} \dots u_1 u_0) \leftarrow (u_{n-1} \dots u_1 u_0) \setminus d$ ▷ Use algorithm 3
34: **end if**