Notes on arithmetic

The 'Babylonians'—that is to say, the people that inhabited what is now southern Iraq—for reasons not entirely clear to us, ued base 60 in scientific calculation. This offers us an excuse to look in general at what arithmetic with an arbitrary base involves. This will add insight to the familiar decimal arithmetic.

1. Representation in base B

The properties of the usual integer division process can be summarized in this:

- *If a is any non-negative integer and B a positive integer then*

$$a = qB + r$$

where $q \geq 0$ and $0 \leq r < B$. They are both uniquely determined by a and B.

The integer q, which is equal to $\lfloor a/B \rfloor$ is called the **quotient** and r the **remainder**. Here $\lfloor x \rfloor$ is the largest integer equal to or less than x.

Proof. Existence of q and r by mathematical induction. It is trivially true for $a = 0$. Assume it to be true for $b < a$ and let's prove it for $b > 0$. If $a < B$ then we set $q = 0$ and $r = a$. Otherwise $a_* = a - B \geq 0$, and we can apply the induction hypothesis to it to get $a_* = q_* B + r_*$. But then

$$a = a_* + B = (q_* + 1)B + r_* \ .$$

As for uniqueness, suppose

$$a = q_1 B + r_1 = q_2 B + r_2 \ .$$

with $0 \leq r_i < B$. Then

$$(q_1 - q_2)B = r_2 - r_1 \ .$$

But if $q_1 \neq q_2$ the left hand side is a multiple of B while the right hand is less in absolute value than B. This is a contradiction.

Now suppose $B \geq 2$.

- *Every positive integer a may be expressed uniquely as a sum of powers of B*

$$a = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \cdots + a_0 = a_{n-1} : \ldots : a_0$$

where each a_k is an integer with $0 \leq a_k < B$.

Exercise 1. *Prove this. Hint: the principal point is that in this representation of a the low order digit a_0 is teh remainder when a is divided by B, and*

$$a_{n-1}B^{n-2} + a_{n-2}B^{n-3} + \cdots + a_1$$

is the quotient.

Exercise 2. *Explain why this is false for $B = 1$. Where does the proof go wrong?*

Example. Suppose $B = 60$, $a = 15,214$ (in decimal). If we divide a by 60 we get

$$15,214 = 60 \cdot 253 + 34$$

so $a_0 = 34$. Now do it again:

$$253 = 60 \cdot 4 + 13$$

so $a_1 = 13$. Finally, $a_2 = 4$. So in base 60

$$15,214 = 4 : 13 : 34 \ .$$

Exercise 3. *Find the base 60 representation of $1,000,301$.*

Exercise 4. *Write out explicitly the complete algorithm to find the base B representation of an integer a.*

2. Conversion to decimal

There is a very efficient way to find the decimal expression for a number in base B. Suppose the base B expression is $b_{n-1} : \ldots : b_0$. This is

$$b_{n-1}B^{n-1} + b_{n-2}B^{n-2} + \cdots + b_1 B + b_0$$

Of course you can just compute all those powers of B and multiply and add. But it is easy to get a little lost if you do that. A better way is suggested by this sequence of calculations for the example $17 : 23 : 31 : 9$ in base 60, which is called **Horner's method**:

$$\begin{aligned} &17 \\ &17 \cdot 60 + 23 = 1,043 \\ &1,043 \cdot 60 + 31 = 62,611 \\ &62,611 \cdot 60 + 9 = 3,756,669 \end{aligned}$$

Exercise 5. *Use Horner's method to find the decimal expression for* $31 : 19 : 41 : 43$.

3. Addition

How to add two numbers expressed in base B?

Let's do an example.

	1	1: 1	1: 1: 1	1: 1: 1
17:34:20:47	17:34:20:47	17:34:20:47	17:34:20:47	17:34:20:47
+ 29:51:29	+ 29:51:29	+ 29:51:29	+ 29:51:29	+ 29:51:29
	16	12:16	4:12:16	18: 4:12:16

We work from right to left, and start by adding 47 and 29 to get (in base 60) $1 : 16$. We write 16 and carry 1. Etc.

Now suppose more generally that a and b are two non-negative integers. Then we can find the base B expression for $a + b$ with the following algorithm. The state of the algorithm at any moment is determined by the current carry ε and the index k of the current 'digits' we are looking at. To make the algorithm a bit simpler, we use a carry right from the beginning, but of course it begins as 0.

(1) Initialize ε to 0 and k to 0. Pad one of the numbers if necessary by zeroes to make them the same length n. (This simplifies the algorithm somewhat.)

(2) Set $c_k = a_k + b_k + \varepsilon$. If $0 \leq c_k < B$, set the answer's digit $d_k = c_k$, and ε equal to 0. Otherwise set $d_k = c_k - B$, the carry equal to 1, and increment k to $k + 1$.

(3) If $k < n$, go back to (2), otherwise set d_n equal to the final carry and stop. The answer is

$$d_n : d_{n-1} : \ldots : d_0 \ .$$

In other words, we are claiming that the d_k are the digits of the sum $a + b$. This means neither more nor less than that

$$\sum d_k B^k = a + b$$

and also that

$$0 \leq d_k < B \ .$$

To prove the second, for example, we work by induction on k. The induction step is actually the combination of two claims: (i) that $d_k < B$ and (ii) the carry is always at most 1.

It is probably worthwhile to formalize the logic. The algorithm consists of one basic step repeated several times, and that operation is the addition of two single digit numbers together with a carry.

- *If* $0 \leq a, b < B$, $0 \leq \varepsilon \leq 1$ *and*

$$c = a + b + \varepsilon$$

then the base B *expression for* c *is* $\varepsilon : d$ *where*

$$d = c, \quad \varepsilon = 0$$

if $c < B$ *and*

$$d = c - B, \quad \varepsilon = 1$$

if $c \geq B$.

The first case is clear, and the second case is valid because in these circumstances we have also $a, b \leq B - 1$ and $B \leq c \leq 2B - 1$. This is used in an argument mathematical induction on k.

As for verifying that the answer actually represents the sum, this also goes by induction. The induction step is that at each step in (2)

$$\varepsilon : d_k : \ldots : d_0$$

represents the sum of $a_k : \ldots : a_0$ and $b_k : \ldots : b_0$.

Exercise 6. *Put in all the details. Try to prove by induction all at once that* $\varepsilon : d_k : \ldots : d_0$ *is the base* B *expression for the* k*-th sum, and that* ε *is either* 0 *or* 1.

4. Subtraction

This is left as an exercise.

Exercise 7. *Describe the algorithm for subtraction.*

Exercise 8. *Prove that it is correct.*

The way to start is to formulate the basic step as we did for addition.

5. Multiplication

Let's do a simple example in base 60 to recall the difficulties. What is 43 : 17 : 7 times 20 : 23?

	2	6: 2	16: 6: 2	16: 6: 2
43:17: 7	43:17: 7	43:17: 7	43:17: 7	43:17: 7
× 20:23	× 20:23	× 20:23	× 20:23	× 20:23
	41	33:41	35:33:41	16:35:33:41

We work row by row, each row corresponding to a different digit on the bottom. Start by calculating $23 \cdot 7 = 2\,41$. So we write 41 and carry 2. Then we multiply $23 \cdot 17 + 2 = 6\,33$. Etc. Then we do the second row, shifted left. Then finally we add the two rows.

$$\begin{array}{r} 15{:}\ 5{:}\ 2 \\ 16{:}\ 6{:}\ 2 \\ 43{:}17{:}\ 7 \\ \times\quad 20{:}23 \\ \hline 16{:}35{:}33{:}41 \\ \hline 14{:}25{:}40{:}20 \\ \hline 14{:}42{:}15{:}53{:}41 \end{array}$$

The basic step here seems to be to find the base B representation for

$$c = a \cdot b + \varepsilon$$

where $0 \leq a, b, < B$ and in addition there is some condition on ε. What condition? The largest product possible here is $(B-1)(B-1)$. We calculate If we multiply

$$(B-1)\cdot(B-1) = B^2 - 2B + 1 = (B-2)B + 1 = B-2:1$$

so we shall impose the condition $\varepsilon \leq B-2$.

We'll see the algorithm for multiplication in two steps. The first is **short multiplication**, of a single digit number a by an arbitrary one $b_{n-1} : \ldots : b_0$.

(1) Initialize the carry ε to be 0, $k = 0$.

(2) Multiply $a \cdot b_k + \varepsilon = c_k$. Divide by B to get $c_k = B \cdot q + d_k$. Set $\varepsilon := q$, increment k.

(3) If $k < n$, go to (2). Otherwise set $d_n = \varepsilon$ and stop. The answer is $d_n : d_{n-1} : \ldots : d_0$.

Multiplication of larger numbers is more complicated. You can just do all the lower digits one by one and then add all the shifted rows, as suggested above. But there is a cleverer way, which exhibits a beautiful fact. In this procedure, the basic step is somewhat more complicated, amounting to adding together a product, a single digit, a multiplicative carry, and an additive carry, all in one operation. The miracle is that it is feasible.

- *If* $0 \leq a, b, c < B$, $0 \leq \varepsilon_a \leq 1$, *and* $0 \leq \varepsilon_m \leq B-2$ *then*

$$0 \leq a \cdot b + c + \varepsilon_a + \varepsilon_m \leq (B-1)B + (B-1) = B-1 : B-1\ .$$

In other words, the sum is at most a two-digit number, and the total carry for a subsequent step will be at most $B-1$.

The algorithm for calculating the product $c_{m+n-1} : \ldots : c_0$ of $a_{n-1} : \ldots : a_0$ and $b_{m-1} : \ldots : b_0$ now becomes:

For $i = 0$ to $m+n-1$ set $c_i := 0$
For $i = 0$ to $n-1$
  $\varepsilon := 0$
  For $j = 0$ to $m-1$
    Set $x_{i+j} := a_i b_j + c_{i+j} + \varepsilon$
    Let $\varepsilon : c_{i+j}$ be the representation of x_{i+j}
  Set $c_{i+m} := \varepsilon$

Exercise 9. *Express* $10,000,000$ *and* $999,999$ *in base* 60 *and multiply them in this way.*

6. Division

Division, as you will recall from elementary school, is much more difficult than the previous algorithms, since it requires some trial and error. I am going to explain how to reduce the guesswork involved, but even the most efficient version will still involve some trial and error.

Short division, at least, is quite simple. Suppose we want to divide $b_{n-1} : \ldots : b_0$ by a single digit number a to get the quotient and remainder.

(1) Initialize the carry, setting $\varepsilon := 0$.

(2) For $i = 0$ to $n-1$ divide to get $\varepsilon : b_i = q \cdot a + r$, set $c_i := q$, new $\varepsilon := r$.

(3) The quotient is $c_{n-1} : \ldots : c_0$ and remainder ε.

In doing this, it is a good idea to show the initial carry by tacking on a '0' at the left hand end of the dividend.

$$\begin{array}{r} 0{:}28{:}12{:}11 \\ 37\overline{)\ 0{:}17{:}23{:}31{:}\ 9} \\ \underline{17{:}16\phantom{{:}31{:}\ 9}} \\ 7{:}31\phantom{{:}\ 9} \\ \underline{7{:}24\phantom{{:}\ 9}} \\ 7{:}\ 9 \\ \underline{6{:}47} \\ 22 \end{array}$$

The basic operation here is to divide a two digit number by a one-digit number, with the condition that the quotient is a one-digit number. For example, in this example I converted $17:23$ (base 60) to $1,043$ (base 10), divided it by 37 get quotient 28, remainder 7. In this 'very short' division, the remainder has to be a one-digit number because it is less than the divisor.

$$\begin{array}{r} 28 \\ 37\overline{)17{:}23} \\ \underline{17{:}16} \\ 7 \end{array}$$

This is in some sense the inverse of multiplying two one-digit numbers to get a two-digit number, and normally involves some special way to handle it—maybe memorization (of the base 60 multiplication tables!) or a lookup. At the end of each 'very short' division, the remainder becomes the next carry in the longer procedure. Etc.

Long division—when the divisor has more than one digit—is much more complicated. You calculate the digits of the quotient one by one, but in each stage there is some uncertainty. When you learned how to do it in elementary school, you probably were taught to guess the quotient digits in a somewhat random fashion. This might have seemed unsatisfactory, but it turns out that even in the most scientific method I know of there is a certain amount of trial and error required. The science lies in reducing this to a minimum.

The basic idea is to get one digit of the quotient at a time, so the basic operation is to divide a number of $n+1$ digits by one of n digits, under the assumption that the quotient be of one digit. This condition may be unsatisfied at the first step, and can be guaranteed even there by tacking on a zero at the beginning. After the first step it will always be OK.

$$16{:}31{:}27\overline{)15{:}17{:}11{:}31} \qquad\qquad \begin{array}{r} 57 \\ 16{:}31{:}27\overline{)15{:}17{:}11{:}31} \end{array}$$

In dividing $15:17:11:31$ by $16:31:27$ the only obvious way to guess the quotient digit is by looking at the first digit of the divisor and the first two of the dividend. Here we divide $15:17$ by 16, getting a quotient 57.

$$\begin{array}{r} 57 \\ 16{:}31{:}27\overline{)15{:}17{:}11{:}31} \\ \underline{15{:}41{:}52{:}39} \end{array} \qquad \begin{array}{r} 56 \\ 16{:}31{:}27\overline{)15{:}17{:}11{:}31} \\ \underline{15{:}25{:}21{:}12} \end{array} \qquad \begin{array}{r} 55 \\ 16{:}31{:}27\overline{)15{:}17{:}11{:}31} \\ \underline{15{:}\ 8{:}49{:}45} \\ 8{:}21{:}46 \end{array}$$

This turns out to be to large, so we decrease it to 56. Still too large, try 55, which works.

There is a problem, however—you may have to decrease the trial quotient a huge number of times. Try following these rules strictly with

$$1{:}59{:}59\overline{)\ 1{:}\ 0{:}\ 0{:}\ 0}$$

So following these rules leads to an answer, but might involve a huge amount of work. There are one or two tricks in the bag left to try out.

(1) **Normalize the divisor and dividend**. Keep doubling both divisor and dividend until the divisor has as its first digit something at least as large as $B/2$. For example, in the example above, we would replace $1:59:59$ by $16\cdot 1:59:59 = 31:43:44$, and the $1:0:0:0$ by $16:0:0:0$.

$$31{:}43{:}44\overline{)16{:}\ 0{:}\ 0{:}\ 0}$$

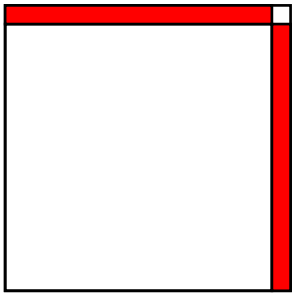
Of course we are changing the problem. However, in multiplying the divisor and dividend by a constant C we are not changing the quotient, and multiplying the remainder by C as well. So if we do the normalized problem we just finish up by dividing the remainder we get by C. The great advantage of normalization is that any trial quotient we get is guaranteed to be no more than 2 too big.

Exercise 10. *Find the quotient and remainder in this example, using normalization.*

(2) When checking a trial quotient, stop short at checking to see whether it works for $a_{m-1}:a_{m-2}$ divided into $b_{n-1}:b_{n-2}:b_{n-3}$ (2-digit into 3-digit division). This will eliminate nearly all oversize trial quotients without an inordinate amount of work.

7. Square roots

Here we really need only an approximation formula. The diagram



leads to

$$\sqrt{N^2+n} \sim N + \frac{n}{2N}$$

There is another algorithm that will prove useful later on. This one finds $\lfloor\sqrt{N}\rfloor$.

(1) Start by setting $x = 1$.

(2) Set

$$x := \left\lfloor \frac{x + N/x}{2} \right\rfloor .$$

(3) If $x > N/x$ then go to (2). Otherwise stop.

Exercise 11. *Prove that this process does indeed end with* $x = \lfloor\sqrt{N}\rfloor$.