

**Q: Show that every positive integer has a multiple whose decimal representation contains only the digits 0 and 1.**

We will call these curiosities ‘1-0’ numbers. This note gives two types of response to the question: first, an existence theorem, which proves the proposition without giving much insight into the nature of the multipliers, and second an algorithm for computing an arbitrary number of multipliers  $k$  for a given integer  $q$  such that  $z = kq$  is a 1-0 number.

**1 Existence of  $k$**

The proof of the existence of some suitable  $k$  for each  $q$  uses the pigeon hole principle and the Euclidean division algorithm for integers. The latter states that, given  $z$  and  $q$ , there are unique integers  $k$  and  $r$  (multiplier and remainder) such that  $z = kq + r$  where  $r$  is strictly less than  $q$  (this is the key point). This can also be written  $z \equiv r \pmod q$ . There are  $q$  values of  $r$ , viz.  $0, 1, 2, \dots, q-1$ .

We want  $z$  to be a 1-0 number. Consider the infinite set of repunits  $\{1, 11, 111, 1111, \dots\}$ . If any one of these is divisible by  $q$  (e.g.  $3 \mid 111, 17 \mid 1111111111111111$ ), we have a 1-0 number, admittedly consisting entirely of ‘1’s. If it is not divisible by  $q$ , there is a non-zero remainder less than  $q$ . By the pigeon hole principle, if we take  $q$  such repunits from the above set, at least two of them,  $z_1, z_2, z_1 > z_2$ , must have the same remainder. Therefore their difference  $z_1 - z_2 \equiv 0 \pmod q$ . This difference will be a 1-0 integer with  $z_1 - z_2$  ‘1’s on the left and  $z_2$  ‘0’s on the right (units end).

As an example,  $111111111111 \equiv 3 \pmod 8$  and  $111 \equiv 3 \pmod 8$  so  $8 \mid 111111111000$ .

The above argument is quite general; it does not depend on  $q$  being prime or having any other special feature. The argument therefore proves the existence of a 1-0 multiple of any given  $q$ , though without showing how  $z_1, z_2$  should be chosen. However, the 1-0 number can be quite large – up to  $q$  digits in the case of some primes such as 3, or  $q - 1$  digits such as 17 (see above). Furthermore, it always has the same form, with leading ‘1’s and (possibly) some following ‘0’s. We consider below a method for finding smaller  $k$ , and for creating 1-0 numbers of almost arbitrary length and distribution of ‘0’s and ‘1’s.

In passing, note that by multiplying the 1-0 number by any integer  $< 10$  we can create ‘2-0’, ‘3-0’, ... numbers. Thus  $8 \mid 55555555000$ .

**2 Algorithm for determining multipliers  $k$**

Suppose  $k$  is  $a_n \dots a_2 a_1 a_0$  where the  $\{a_j\}$  represent digits.  $k$  will be constructed sequentially from the right (units) end. Three parts to developing an algorithm are:

1. proving that the rightmost digit  $a_0$  can always be found.
2. proving the successive digits  $a_1, a_2, \dots$  can be concatenated on the left
3. presenting evidence that  $k$  can always be terminated by determining  $a_n, a_{n-1}, \dots$

We first explain the algorithm by giving two examples, then in Section 2 provide justification for the above.

**1.1 Example 1:  $q = 3$**

The multiplication table for 3 is

0	1	2	3	4	5	6	7	8	9
0	3	6	9	12	15	18	21	24	27

so the rightmost digit,  $a_0$ , in  $k$  can only be 7 (or, trivially, 0). This leaves 2 to carry. Add this 2 to each entry in row 2 of the multiplication table and observe that the entry under 3 becomes 11. Hence select  $a_1 = 3$ , and the product is 111, a 1-0 number. The algorithm therefore terminates, giving  $k = 37$  and  $kq = 37 \times 3 = 111$ . This is the basis for more complex cases, as described below.

### 1.2 Example 2: $q = 62$

The table below shows sequential stages in the algorithm. The central two rows marked  $a_j$  and T give the 62 times table. The rows below correspond to sequential stages in determining  $k$ , one digit per row. The rows above list the ‘exits’ from the algorithm; these are the carry values which can cause a 1-0 number to be created at the next stage. The possible 1-0 digit sequences for 62 are listed in the first column, and the corresponding terminating carry values in the column above ‘62 times 1’. These are the differences of the multiples of 62 from the 1-0 numbers listed to the left. The fact that all of these terminating carry values are in this column means that the last (leftmost) digit in  $k$  must be ‘1’ in this case of  $q = 62$ .

111	49										
110	48										
101	39										
100	38										
11											
1											
$a_j$	0	1	2	3	4	5	6	7	8	9	
T	0	62	124	186	248	<b>310</b>	372	434	496	558	5
+ 31	<b>31</b>	93	155	217	279	341	403	465	527	589	0
+ 3	3	65	127	189	<b>251</b>	313	375	437	499	561	4
+ 25	25	87	149	<b>211</b>	273	335	397	459	521	583	3
+ 21	21	83	145	207	269	<b>331</b>	393	455	517	579	5
+ 33	33	95	157	219	<b>281</b>	343	405	467	529	591	4
+ 28	28	<b>90</b>	152	214	276	338	400	462	524	586	1
+ 9	9	71	133	195	257	319	<b>381</b>	443	505	567	6
+ 38	38	<b>100</b>	162	224	286	348	410	472	534	596	1

The algorithm proceeds as follows. Inspection of the 62 times table shows that only  $5 \times 62 = 310$  ends in either ‘0’ or ‘1’, so  $a_0$  can only be 0. The selection of 310 is highlighted in bold and 5 recorded in the rightmost column. There is 310 to carry to the calculation of  $a_1$ , so the next row down shows 310/10 added to row T. Here are two choices for  $a_1$  – either 0 or 4. We chose 0, as indicated again in bold and by recording ‘0’ in the rightmost column. 300 are now carried forward to the calculation of  $a_2$  and so 3 is added to row T, giving the next row down. The method is repeated until a terminating carry value is occurs – in this case 38. At the final stage  $38+62 = 100$  and we can stop. The value of  $k$  created is read from the rightmost column.  $62 \times 161453405 = 10010111110$ .

### 1.3 Numbers ending in 5

There are alternative ways to deal with these numbers, but it is probably simplest to divide them by 5 and find  $k$  for the quotient. For example,  $155 \times 7162 = 1110110$  so  $155/5 \times 7162/2 = 111011$ . From the algorithm we find  $31 \times 3581 = 111011$  and from this  $k$  for 155 is readily obtained.

### 1.4 Multiple solutions and solutions of arbitrary length

In many cases there is a choice of which next digit  $a_{j+1}$  to select in the construction of  $k$ , because more than one digit will give a ‘0’ or a ‘1’. It is likely that with appropriate selection, termination of the algorithm can be avoided indefinitely to produce values of  $k$  of great length for any given  $q$ . For instance,

$$\begin{aligned}
7 \times 143 &= 1001 \\
7 \times 15873 &= 111111 \\
7 \times 1428573 &= 10000011 \\
7 \times 15873015873 &= 111111111111 \\
7 \times 1444444444428573 &= 10111111111000011
\end{aligned}$$

Regarding termination of the algorithm, observe what happens to the product  $kq$  as the algorithm is continued. If  $q$  has  $N$  digits, the effect is to push a sequence of  $N$  digits ( $N-1$  if  $q$  ends in 5) to the left, while filling in the rest of the number with an increasing 1-0 train on the right. This is illustrated below for  $q = 250505$  having  $N = 6$ . The five-digit sequence is shown in bold.

- $17822 \times 250505 = \mathbf{44645}00110$
- $122417117822 \times 250505 = \mathbf{30666}100100000110$
- $27367122417117822 \times 250505 = \mathbf{68556}01001100100000110$
- $51839684277758927367122417117822 \times 250505 = \mathbf{12986}100110000000100101001100100000110$
- $\mathbf{32285}1839684277758927367122417117822 \times 250505 = 80876000100110000000100101001100100000110$
- $979545322851839684277758927367122417117822 \times 250505 = \mathbf{24538}1001101000100110000000100101001100100000110$

The behaviour is reminiscent of a pseudo-random number generator; we hypothesise that long sequences of 5 digits can be generated by this process, and this is sufficient to ensure that it will terminate on one of the ‘available’ 1-0 five digit sequences. This is explored further in section 2.3 below.

## 2 Justification of key stages

### 2.1 Rightmost digit $a_0$

$a_0 = k \bmod 10$  and it is necessary to be assured that  $a_0$  can be chosen so that  $kq \bmod 10$  is 0 or 1. The table below gives  $a_j q \bmod 10$  for all values of  $q \bmod 10$  and  $a_j = 1$  to 10. It shows that there is at least one 0 or 1 in each row so  $a_0$  can be selected whatever the value of  $q$ .

		columns list $a_j$								
		<b>1</b>	2	3	4	5	6	7	8	9
		2	4	6	8	<b>0</b>	2	4	6	8
	rows	3	6	9	2	5	8	<b>1</b>	4	7
	list	4	8	2	6	<b>0</b>	4	8	2	6
	$q \bmod 10$	5	<b>0</b>	5	<b>0</b>	5	<b>0</b>	5	<b>0</b>	5
		6	2	8	4	<b>0</b>	6	2	8	4
		7	4	<b>1</b>	8	5	2	9	6	3
		8	6	4	2	<b>0</b>	8	6	4	2
		9	8	7	6	5	4	3	2	<b>1</b>

### 2.2 Middle digits

In general there will be a non-zero value  $c_j = (qa_j \text{ div } 10^j)$  to carry forward to the calculation of  $a_{j+1}$ . This carry operation is equivalent to adding  $c_j$  to each row in the above table. For those integers  $\{1, 3, 7, 9\}$  which are coprime to 10, all integers 1 to 9 occur in the row, so adding any value  $c_j$  is

simply equivalent to permuting the row entries and allows us still to choose  $a_{j+1}$  so that the next digit in  $kq$  is a '1'.

For rows 2, 4, 6, 8 above, adding an even value of  $c_j$  merely permutes the row entries, whilst adding an odd  $c_j$  will produce one entry of '1'. Again  $a_{j+1}$  can be chosen.

Regarding row 5,  $q$  numbers ending with '5' require either:

- circumvention by dividing by 5, as described in Section 1.3. above, or
- taking a *pair* of digits at a time. (Details omitted.)

Taken together the arguments for the sets  $\{1, 3, 7, 9\}$ ,  $\{0, 2, 4, 6, 8, \}$  and  $\{0, 5\}$  of values of the digits in  $q$  prove that the next digit,  $a_j$ , in the multiplier  $k$  can always be chosen. Thus the algorithm can be continued indefinitely.

### 2.3 Termination

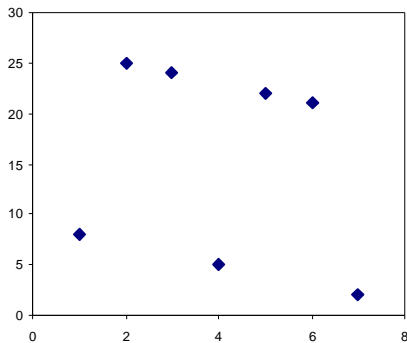
There are two ways to look at termination of the sequence which represents  $k$ :

- as in Section 1.4, as a string of  $N$  random digits being 'pushed to the left' by an increasing train tail of '0's and '1's. The algorithm terminates when this string consists of only '0's and '1's.
- as the occurrence of a 'terminating carry value', which are those differences, less than  $q$ , between the multiples of  $q$  and those 1-0 numbers greater than  $q$ .

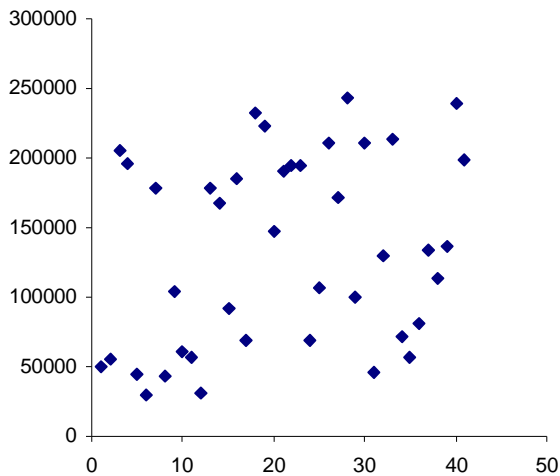
We have already noted the similarity of the  $N$ -digit string in a) and the sequence of carry values to a pseudo-random number generator. The author confesses to being unable to provide a proof that a sufficiently long sequence of carry values in b) will be generated such that a terminating value will always occur (or equivalently, that a sufficiently long sequence of pseudo-random digits in a) will be produced for a 1-0 sequence always to occur.) However, there are three items of evidence suggesting that this is a not implausible hypothesis:

- For about 20 values of  $q$  examined, termination of the algorithm has been possible after only a few steps, corresponding to a relatively small value of  $k$ . One observes that the carry values occur in interwoven sequences, each sequence descending and eventually reaching one of the terminating values. The following two plots illustrate the trends.

Carry values for  $q = 27$ , illustrating the tendency to cluster in falling sequences.



Carry values for  $q = 25025$ , illustrating the pseudo-random distribution, but with trends discernible.



- 2) Consider the availability of terminating carry values.  $2^n$  1-0 numbers occur between every  $10^n$  and  $10^{n+1}$  for any  $n$ . Since  $a_{j-1}$  can be chosen in the range 0 to 9, and carry values  $c_j$  of up to  $q-1$  can occur, the number  $(a_j \cdot q + c_j)$  can span from 0 to  $10q - 1$ . Thus for any  $q$  typically  $2^n$  terminating carry values will be available. This means that the length of any pseudo-random sequence only need be  $q \cdot 2^{-n}$ .
- 3) The (supposed) pseudo-random number generator for carry values  $c_j$  has recursion relation  $c_{j+1} = a_j \cdot q + c_j \text{ div } 10$  where  $a_j$  is the (variable) multiplier chosen to obtain a '0' or '1' in  $k$ , and 'div' means the integer quotient (as opposed to 'mod'). Compare this with the forms of two computing standard number generators:
  - i) the multiplicative linear congruence generator:
 
$$c_{j+1} = a \cdot c_j + b \text{ mod } n$$
 in which  $a$  is a fixed multiplier,  $b$  a fixed additive term, and  $n$  a large number coprime to  $a$  and  $b$ .
  - ii) the subtract-with-borrow generator used in Matlab because of its exceptionally long sequence.
 
$$c_{j+1} = c_{j-r} - c_{j-s} - b \text{ mod } n$$
 in which  $b$  is a 'borrow' bit set to 0 or 1 according to whether  $c_{j-r} - c_{j-s}$  is  $\leq 0$  or  $> 0$ . This shares with the recursion relation of our algorithm the property that one of the parameters is adjusted for each step  $j$ .

We suggest that the similarities are sufficient to consider the carry values as a pseudo-random sequence. Determination of the length of the cycle would require further investigation.

**Reference**

“On the Period Length of Pseudorandom Number Sequences” A. Glen, University of Adelaide, 2002