

## 16. Factoring Numbers

We will describe several methods for factoring. The first of these is kind of fun, but not the best possible, since it becomes hopeless to use if for factoring numbers above say  $10^{50}$  and probably less; it takes a number of steps of the order of  $N^{1/4}$  to factor  $N$ , and the steps are not very simple, though not hard in principle. This method is based on iterating a function, mod  $N$ .

There are two more serious methods which we will discuss some in the next section. One, the more prosaic of the two, is called the quadratic sieve. The other is based on raising an initial element of a strange group to increasing factorial powers in that group, mod  $N$ . The group is that associated with an elliptic curve, as we shall see. Strangely enough these methods have similar expected performance, but the quadratic sieve is easier to apportion to a network of different computers.

### 1. The Function Iteration or Cycle Method for Factoring

Suppose  $N=pq$ , and we seek to factor  $N$ .

The idea of the method is we will choose a random starting integer and choose a polynomial function of it. We will then iterate this function.

Suppose you iterate a polynomial function, which means start with  $x$ , form  $f(x)$  then  $f(f(x))$  and  $f(f(f(x)))$  and so on (wonderfully well adapted to do on a spreadsheet!) and you do it mod  $N$ . By the Chinese Remainder theorem you can think of what is happening mod  $p$  and mod  $q$  separately.

And what will happen, say, mod  $p$ ? Well you will start from  $x$ , move to say,  $x_1$ , then  $x_2$  and so on, and so wander around mod  $p$ , for a while what seems like randomly. Then at some point you will reach a value,  $x_j$  you had reached previously, and then you will cycle around following your previous itinerary, because once you are at a point you had previously visited, your next destination will be the same as it was last time you were there. (our wandering seems random but is actually completely determined by our function  $f$ .)

This raises two questions. How long will you have to wander until this happens, that is, until you start cycling?

And how can we use this nonsense to factor  $N$ ?

The answer to the first question is we can expect to return to a previously attained value

in a number of steps that is of the order of  $(p)^{1/2}$ . (Why? Suppose we were actually wandering at random, so that at each step we have probability  $1/p$  of reaching any value mod  $p$ . We could then compute the probability that we do not hit a value we had reached before.

After  $k$  steps it will be

$$\prod_{j=1}^k (1-(j/p)).$$

Because if we have not already done so, after  $k-1$  steps we will have been at  $k$  different places out of a total of  $p$  of them.

This is exactly the product we encounter with the famous 'birthday surprise' and by taking logs and expanding we can find it looks like

$$\exp(-k*(k-1)/(2*p)).$$

So within a number of steps of at most the order of  $k^2 \ln(k)/p$  it will become highly probable that we will have reached a place we had visited already.

And now what can we do with this fact?

We will create a tortoise and a hare. They will start from the same starting point mod  $N$ . The tortoise will iterate the function once on each turn. The hare will iterate it twice on each turn. And now observe that once they are both inside the cycle, the hare will move one closer to the tortoise on each step, and will catch up with it at some step, mod  $p$ .

And then we have a winner! What we can do is at each step, look at the difference between the hare's value and the tortoise's value, and apply Euclid's algorithm to  $N$  and that difference.

When the hare and the tortoise are at the same value mod  $p$  (and not mod  $q$ ) the difference will be  $0 \bmod p$  (but not mod  $N$ ), and will therefore be a multiple of  $p$ . When we find its gcd with  $N$ , which Euclid's Algorithm will give us, we will find  $p$ !

Can this procedure fail? Yes, in two ways. One, we could get a tortoise-hare difference of  $0 \bmod p$  and  $0 \bmod q$  at exactly the same step, in which case we get nothing. Or we could be very unlucky and have to iterate much more than we expect to get equality mod  $p$ .

If either of these things seem to be happening, we can choose a different start or a different function to iterate.

Is all this doable? It is practically trivial to iterate a function on a spreadsheet. Euclid's algorithm is easy to implement also, If the two starts are next to each other with the bigger one to the left you can enter to the right of the right one  $=\text{mod}(\text{one on left}, \text{one on right})$  and copy

to the right a healthy distance. At some start you will get to 0, and the gcd you seek will be the entry just before you get to 0. So enter N to the left, and your hare tortoise difference on the right and copy this until you find the gcd.

Let's do an example

For function we try  $x^2 + x + 1$ . For start we choose 1

Now let us pick a number to factor. Say 4097003. Enter it in A1

We put our start say in A2 and in B2 put =A2

In c2 we put =mod(b2\*b2+b2+1,\$a\$1) and in A3 put =mod(a2\*a2+a2+1,\$a\$1) and in b3 put =mod(c2\*c2+c2+1,\$a%1)

Now we copy a3 and b3 down a couple of hundred rows, and do the same for c2.

Notice that column A iterates our function once in each row, while column B (or C) iterates twice in each row.

Now if in column D everywhere we put =A\$1 and in E2 (and copy down) put =abs(b2-a2) we are ready to to start Euclid's algorithm in the columns f g h and so on.

This can be accomplished with one instruction, put in f2 and copied down and to the right, as far as needed. The instruction is =mod(d2,e2).

What will happen is we will Euclid along each row until we hit 0 after which we get crap from the spreadsheet. However the entry we get just before we get 0 is the gcd of our hare-tortoise difference and N.

When hare has caught up to tortoise mod p and not mod q this will spit out a factor of N.

Notice that we can change our start by changing A2 and can try to factor a different N simply by changing A1.

Here is an example worked out though with different columns. The function used is  $x^2+x+1$  and the start is 11. We have cut off the spreadsheet on the right so that it fits on the page.

Notice that 89 appears in the 8<sup>th</sup> row counting from the start, and 139 in the 12<sup>th</sup>.

A	B	C	D	E	F	G	H	I	J	K
12371	$x^2+x+1$									
11	11	133								
133	5452	2244	12371	5319	1733	120	53	14	11	3
5452	2784	9195	12371	2668	1699	969	730	239	13	5

2244	1436	9947	12371	808	251	55	31	24	7	3
2784	9499	6427	12371	6715	5656	1059	361	337	24	1
9195	5988	10975	12371	3207	2750	457	8	1	0	#DIV/0!
1436	5174	4607	12371	3738	1157	267	89	0	#DIV/0!	#DIV/0!
9947	421	4469	12371	9526	2845	991	863	128	95	33
9499	9637	12310	12371	138	89	49	40	9	4	1
6427	3661	8790	12371	2766	1307	152	91	61	30	1
5988	3625	6249	12371	2363	556	139	0	#DIV/0!	#DIV/0!	#DIV/0!
10975	1004	6970	12371	9971	2400	371	174	23	13	10
5174	6954	6832	12371	1780	1691	89	0	#DIV/0!	#DIV/0!	#DIV/0!

If you want to avoid all the DIV/0!'s, you can add an if to the statement implementing Euclid's algorithm: `if(e2>0,mod(d2,e2),0)`. This will cause the row to alternate 0's and the gcd.

This method has the drawback that it expects to take time of the order of  $N^{1/4}$  when N is the product of two roughly equal primes. This is not good enough for modern applications.