

Computer Tutorial 8

Start MAGMA and (as usual) begin with a `SetLogFile(...)` command.

1. Define P to be the set of polynomials in x over the integers via the command

```
P<x>:=PolynomialRing(Integers());
```

Then try some calculations, e.g. `f:=x^5-2*x^4-8*x^2-x-6; g:=x^2+1; g^3; g^50; g^6+f; Factorization(f); g^10 mod f;`

2. Here is a MAGMA function that tries to factorize an integer using Pollard's Rho method.

```
PRho:=function(n,f,s)
  a:=s; b:=s; i:=1;
  repeat
    a:=Evaluate(f,a) mod n;
    b:=Evaluate(f,b) mod n;
    b:=Evaluate(f,b) mod n;
    g:=GCD(b-a,n);
    "a[" ,i," ] is",a;
    "a[" ,2*i," ] is",b;
    "gcd(" ,b,"-",a," ,n) is",g,"
    ";
    i:=i+1;
  until g gt 1;
  "Factorization of",n,"found after",i,"iterations:";
  return [g,n/g]; end function;
```

The argument n is the number to be factorized, f is the polynomial function used (usually $x^2 + 1$) and s the starting value. If a_0, a_1, a_2, \dots is the sequence defined by the rules that a_0 is s and a_i is the residue of $f(a_{i-1}) \bmod n$ for all $i > 0$, then the successive values taken by a are a_0, a_1, a_2, \dots , while the successive values taken by b are a_0, a_2, a_4, \dots . The process stops after i iterations, where i is the least positive integer such that $\gcd(a_{2i} - a_i, n) > 1$.

Test it with `PRho(1001,x^2+1,1);` and `PRho(11111111111,x^2+1,1);`. Then try out `n:=Random(10^30); PRho(n,x^2+1,1);`, and then repeat the following a few times: `p:=RandomPrime(25); q:=RandomPrime(25); PRho(p*q,x^2+1,1);`.

Try the same commands with $x^2 + 1$ replaced by other things, like $x^2 + 3$, $x^2 + 31$, $x^4 + 1$. Also try varying the starting value. (If it ever seems to be taking too long you can interrupt it with Control-C.) To compare times taken by the different variations you can put `time` before the command.

3. Type `load "tut8data.txt";`. This will replace your `PRho` function above by a version that is exactly the same but with lines 9 to 12 removed (so that it doesn't print out so much stuff). It also defines a function `BPRho` as follows:

```
BPRho:=function(n,f,s)
  a:=s; b:=s; i:=1; j:=2;
  repeat
    if i eq j then
      j:=2*j;
      b:=a;
    end if;
    i:=i+1;
    a:=Evaluate(f,a) mod n;
    g:=GCD(a-b,n);
  until g gt 1;
  "Factorization of",n,"found after",i,"iterations:";
  return [g,n/g];
end function;
```

This computes $\gcd(a_j - a_i, n)$ only when i is a power of 2 and $i < j \leq 2i$. It is meant to be faster since each loop of `PRho` involves three evaluations of f and calculation of one gcd, whereas each loop of `BPRho` involves only one evaluation of f and one gcd. However, `BPRho` may need more iterations to complete. Compare the two by repeating the sequence of commands

```
p:=RandomPrime(45); q:=RandomPrime(45); n:=p*q;
time PRho(p*q,x^2+1,1);
time BPRho(p*q,x^2+1,1);
```

a dozen or so times. (You can replace 45 by something smaller if 45 takes too long, or by something bigger to see how much slower it becomes.)

4. If BPRho completes in h steps then PRho will take no more than $2h$ steps. If PRho completes in k steps then BPRho will take no more than $3k - 2$ steps. (Exercise: prove these statements!) But in my experiments I usually found that when PRho takes k steps then BPRho takes *exactly* $2^\ell + k$ steps, where 2^ℓ is the least power of 2 with $2^\ell \geq k$. Check this for the tests you carried out above. (Hint: ℓ is the least integer such that $\ell \geq \log_2(k)$. So, for example, if PRho takes 1260864, use `Log(2,1260864)` to find that ℓ must be 21, and calculate $2^{21} + 1260864$.) Note that each gcd takes significantly longer than each evaluation of f ; so extra gcd calculations could easily more than compensate for fewer evaluations of f .
5. Recall that the n -th repunit is the number $(10^n - 1)/9$ (whose base 10 representation consists of n ones). Using Fermat's Little Theorem to help you, find a repunit that is divisible by the prime 2531, and then find the smallest such repunit .

[Use MAGMA's `IsDivisibleBy` function, but beware of a MAGMA trap. To see it, type `IsDivisibleBy((10^2-1)/9,7);`, and observe that MAGMA says `true` even though 11 is not divisible by 7. This is because MAGMA interprets $(10^2 - 1)/9$ as a rational number rather than an integer. Instead you should use `IsDivisibleBy((10^2-1) div 9,7);` or `IsDivisibleBy(Integers()!((10^2-1)/9),7);`. (If a and b are integers then `div(a,b)` is the integer quotient when a is divided by b . The second method uses MAGMA's "coercion" operator, `!`: the object following the `!` gets converted (if possible) into an element of the set preceding the `!`.)]

6. The MAGMA function `Modorder` returns the order of a modulo n . This quantity, which was denoted by $\text{ord}_n(a)$ in lectures, is the least positive integer m such that $a^m \equiv 1 \pmod{n}$. (Note that a must be coprime to n for this to be defined; `Modorder` will return 0 if $\text{gcd}(a, n) > 1$.)

Use the MAGMA commands

```
n:=73;
for i:=1 to 72 do
  i,"has order",Modorder(i,73),"modulo 73";
end for;
```

to find all the numbers with order 72 modulo 73.

Check that for each divisor d of 72 there are $\phi(d)$ residues mod 73 with order d . (Just count them by hand.)

7. Repeat the commands given in Exercise 6 with other numbers in place of 73, and check that the result is consistent with the Euler-Fermat Theorem.
8. A residue mod 73 with order 72 is called a *primitive root* modulo 73. If r is a primitive root then every nonzero residue a mod 73 can be expressed as $r^i \pmod{73}$ for some $i \in \{0, 1, \dots, 71\}$. The number i is called the *discrete logarithm of a to the base r modulo 73*. (A similar definition applies for any other prime in place of 73.) Choose a primitive root r and get MAGMA to tell you the discrete logarithms to the base r of all the nonzero residues mod 73. To do this, start with the commands

```
F:=FiniteField(73);
r:=F!5;
```

(where I have chosen 5 as the primitive root), and then do the command `Log(r,F!i);` for all values of i from 1 to 72.

9. Actually, MAGMA will choose a primitive root for you if you do not choose one yourself. Do the following commands:

```
F109:=FiniteField(109);
PrimitiveElement(F109);
for a in F109 do
  if a ne 0 then
    "The discrete log of",a,"modulo 109 is",Log(a);
  end if;
end for;
```

Use the `Modexp` command to confirm a few of the answers. (If r is the primitive element then `Log(a)` should equal b if and only if `Modexp(r,b,109)` equals a .)