

Computer Tutorial 7

Start MAGMA and begin with a `SetLogFile(...)` command. Type `load "tut7data.txt";`.

1. Start MAGMA and type `k:=4096;` followed by `time x:=Factorial(k);`. Since we have seen in lectures that computing $n!$ exactly requires $O(n^2(\log n)^2)$ bit operations, doubling n will multiply the number of bit operations involved by 4 or more, and so we would expect the time it takes magma to perform the calculation will also be multiplied by 4 or more. Check this with `time x:=Factorial(2*k);`, `time x:=Factorial(4*k);` `time x:=Factorial(8*k);`, and so on, until it starts taking too long.
2. Computing powers of large numbers is much faster than computing factorials, since you can use successive squaring. For example, 8765^{8192} is much bigger than $8192!$, but is much faster to compute since it is $(((((8765^2)^2)^2)^2)^2)^2$. Check with `time x:=8765^8192;`. (Compare with `time x:=Factorial(2*k);`, done above.)
3. Compute the base 2 logarithm of 512 via `Log(2,512);`. Now recall that the number of digits in the binary representation of n is the least integer greater than $\log_2(n)$, and suppose that we want to find the number of binary digits in $711^{16000000}$. The command `Log(2,711^16000000)` would take too long. Find a better command.
4. Check that $s = 82349$ is prime, via the commands `s:=82349;` `IsPrime(s);`, and then compute the residue of $711^{16000000} \pmod{s}$. You can do this inefficiently via `711^16000000 mod s;` (time this). A much more efficient method is to use the function `Modexp`. For comparison, do `time Modexp(711,16000000,s);`. What is inefficient about the first method, and what do you think the `Modexp` function does to make things faster?
5. To see that `Modexp` really is fast, choose some random numbers of one or two thousand digits, and time it. i.e. do `a:=Random(10^1000,10^2000);`, then `b:=Random(10^1000,10^2000);` and `c:=Random(10^1000,10^2000);`, and then `time Modexp(a,b,c);`.
6. Still on the theme that `Modexp` is fast, get MAGMA to choose a random 500 bit prime r and use `Modexp` to compute the residue of $a^{r-1} \pmod{r}$ for a randomly chosen a . (Use the following commands: `r:=RandomPrime(500);` `a:=2+Random(r-3);` `r;` `a;` `Modexp(a,r-1,r);`.) Repeat a few times.

Most of the remaining questions are unreasonably hard, since they involve writing little MAGMA programs, but the solutions are given. MATH2988 students ought to try to answer the questions before looking up the solutions, but MATH2068 students can just look up the solutions if they prefer. However, it is important that you read the MAGMA code carefully and make sure that you understand what it is doing, rather than just mindlessly running the commands.

The small catch is that the solutions have been enciphered using an RSA cipher. To decipher you will have to use the method of Questions 5 to 7 of last week's tutorial. The enciphering key was (n, e) , where n is the product of two primes p and q . These numbers were all loaded in the file `tut7data.txt`: get MAGMA to print their values just so that you can see them. Then find the appropriate decryption exponent d . You will need to use commands of the form

```
NaiveDecoding([Modexp(m,d,n): m in ciphertext]);
```

where `ciphertext` should be replaced by the appropriate ciphertext for the question you are doing.

7. The arithmetic functions ϕ , τ , σ and μ that have been defined in lectures are all implemented in MAGMA; their names are `EulerPhi`, `NumberOfDivisors`, `SumOfDivisors` and `MoebiusMu`. For each of these we obtained, in lectures, a formula for the value of the function at n (in terms of the prime factorization of n). In this exercise we pretend that these functions are not implemented in MAGMA, and attempt to find MAGMA code of our own to compute their values, using MAGMA's `Factorization` function.

First, it is useful to know that you can easily get MAGMA to find the product of all the numbers in a sequence. Type

```
&*[3,4];
&*[2,5,9,109];
```

and convince yourself that the answers are right. Then calculate 26!:

```
&*[1..26];
```

Recall that the function `Factorization` returns a sequence of pairs giving the prime divisors of an integer and their multiplicities. Type

```
Factorization(720);
a:=Factorization(1003003001);
a;
a[2];
a[2,1];
a[2,2];
a[2,1]^(a[2,2]-1);
```

What will `&*[t[1]^t[2] : t in Factorization(123456789)]`; return? Check it!

Now write a one-line MAGMA command that will compute $\phi(123456789)$. (Enciphered solution: `encipheredphi`.)

Do the same for $\tau(123456789)$, $\sigma(987654321)$, and $\mu(11111111111111)$. (The enciphered solutions are stored as `encipheredtau`, `encipheredsigma` and `encipheredmu`. The last one is the hardest. You can do it using the `Floor` function: `Floor(x)` is the greatest integer less than or equal to x .)

8. Recall that a *repunit* is a positive integer all of whose digits are 1 (in the usual base 10 notation). For want of a better term, let us call a positive integer a *base b repunit* if all its digits are 1 when it is written in base b notation.

Which numbers are base 2 repunits. (Enciphered answer: `mersenne`. This is not intended to be a hard question!)

By imitating the code used for the last question of Computer Tutorial 6 (see the commented log file) find seven prime numbers that are base 3 repunits. (Enciphered answer: `base3repunits`).

9. Recall that a number is called *perfect* if it is equal to the sum of its divisors, excluding itself. That is, n is perfect if $n = \sigma(n) - n$. Two (unequal) positive integers n and m are said to be *amicable* if $m = \sigma(n) - n$ and $n = \sigma(m) - m$. Find the first 25 pairs of amicable numbers. (Enciphered answer: `amicable`).