The University of Sydney
School of Mathematics and Statistics

Computer Tutorial 7 (Week 8)

MATH2068/2988: Number Theory and Cryptography — Semester 2, 2017

Web Page: http://www.maths.usyd.edu.au/u/UG/IM/MATH2068/
Lecturer: Dzmitry Badziahin

1. In MAGMA, if `s` is a set or a sequence then `&+s;` returns the sum of all the terms in `s` and `&*s;` returns their product. Try this out with some commands such as `&*{2,3,3,3,5};`, `&+[1..10];` or `&*[1..2^18] eq Factorial(2^18);`.

2. Recall that if m is a positive integer, the number of bits (binary digits) in the binary representation of m is the smallest integer greater than $\log_2(m)$. The MAGMA command `Log(b,m)` returns the base `b` logarithm of `m`. Using it, compute the exact numbers of bits in the integer $(2^{18})!$.

3. Stirling's approximation for $m!$ is $\sqrt{2\pi m}\,(m/e)^m$. Use the following commands to compute the resulting approximation for $\log_2((2^{18})!)$, and compare with the previous exercise:

```
pi:=Pi(RealField());
exp:=Exp(1);
m:=2^18;
Log(2,Sqrt(2*pi*m)*((m/exp)^m));
```

4. In lectures we showed that if one uses the primary school "long multiplication" algorithm then computing $m!$ requires $O(m^2(\log_2(m))^2)$ bit operations (since you have to do $O(m)$ multiplications of the form $j \times (j-1)!$, where j has $O(\log_2(m))$ bits and $(j-1)!$ has $O(m\log_2(m))$ bits). So if one uses this algorithm, doubling m would multiply the number of bit operations required to compute $m!$ by 4 or more. However, MAGMA uses a faster algorithm for multiplying large numbers. Start with `m:=20000;` and repeat the commands `time x:=Factorial(m);` and `m:=2*m;` enough times to get a sense of the factor by which the time increases when m is doubled.

5. Suppose that we want to find the number of bits in the integer $711^{16000000}$. The command `Log(2,711^16000000);` takes an unnecessarily long time, since it involves actually computing $711^{16000000}$. Find a better command.

6. Computing powers of large numbers is much faster than computing factorials, since you can use successive squaring. For example, 234567^{262144} is much bigger than $(2^{18})!$ (note that $2^{18} = 262144$), but is much faster to compute since it is

$$(((((((((((((((((234567^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2.$$

Compare `time x:=234567^262144;` with `time x:=Factorial(2^18);`.

7. Check that $s = 82349$ is prime, via the commands `s:=82349; IsPrime(s);`. Suppose you want to compute the residue of $711^{16000000}$ (mod 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, try `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?

8. To see that `Modexp` really is fast, choose some random numbers of one or two thousand digits, and time it. Explicitly:

```
a:=Random(10^1000,10^2000);
b:=Random(10^1000,10^2000);
c:=Random(10^1000,10^2000);
time Modexp(a,b,c);
```

9. Still on the theme that `Modexp` is fast, get MAGMA to choose a random 500 bit prime r with the command `r:=RandomPrime(500);`, choose a random a between 2 and $r-1$ with the command `a:=1+Random(r-2);`, and use `Modexp(a,r-1,r);` to compute the residue of $a^{r-1} \pmod{r}$. Repeat a few times.

Now load tut8data.txt. The remaining exercises are more difficult, and you may wish to do just one of them. They all require writing short MAGMA programs. If you don't feel capable of doing this or prefer not to, there is an alternative: you can look up the sample solutions which are included in the file tut8data.txt you have just loaded. However, these solutions have been enciphered using an RSA cipher! The modulus `n`, the two primes `p` and `q` of which it is the product, and the encryption exponent `e` have been defined by the file tut8data.txt you have just loaded. To decipher the solutions, you will need to compute the decryption exponent `d` and then use commands of the form

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

where `ciphertext` should be replaced by the name of the appropriate ciphertext.

***10.** The multiplicative 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.

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

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

Note that `a[2]` means the second term of `a`, and `a[2,1]` means the first component of `a[2]`, and so on. What will `&*[t[1]^t[2] : t in Factorization(123456789)];` return? Check it!

Now write a one-line MAGMA command that will compute $\phi(123456789)$. (The RSA-enciphered solution is stored under the name `encipheredphi`.) Then 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`.)

***11.** 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`).

***12.** 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`).