The University of Sydney
School of Mathematics and Statistics

Computer Tutorial 5 (Week 5)

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

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

In this tutorial we continue our investigation of block transposition ciphers.

Let $M = c_1c_2c_3 \dots c_N$ be a sequence of letters (such as an encoded plaintext message). A *digraph* is a pair of adjacent letters, that is, c_ic_{i+1} for some i. Last week we made use of the "digraph coincidence index", which is the probability that two randomly chosen digraphs are the same. If all digraphs occurred with equal frequency then this number would be $(1/26)^2 \approx 0.0015$, but because some are much more common than others, typical English text gives a much higher value than this. So computing the digraph coincidence index is one possible way to test whether a given sequence of pairs of letters might be a sequence of digraphs extracted from a piece of English text.

There is another quantity that you can also use for this purpose, called the "coincidence discriminant". In a piece of text that is N letters long, there are $N-1$ digraphs. If the digraph xy occurs $K(x, y)$ times then $p(x, y) = K(x, y)/(N - 1)$ is the probability that a randomly chosen digraph is xy. Define $p(x, -)$ to be the sum over all y of $p(x, y)$; then $p(x, -)$ is the probability that a randomly chosen digraph begins with x. (It is the same as $K(x)/(N - 1)$ where $K(x)$ is the number of occurrences of x in the text in positions other than the last one. For reasonably long texts, this will not differ very much from the relative frequency of x as a single letter, because the effect of excluding the last position will be negligible.) Similarly, define $p(-, y)$ to be the sum over all x of $p(x, y)$; then $p(-, y)$ is the probability that a randomly chosen digraph ends with y. We can compare $p(x, y)$ with $p(x, -)p(-, y)$ to see whether or not x and y have a statistical tendency to occur as a digraph. The coincidence discriminant is the sum over all pairs x, y of $(p(x, y) - p(x, -)p(-, y))^2$. For a sequence of letters generated by randomly choosing each successive letter, with all letters always having probability 1/26 of being chosen, one would expect $p(x, y)$ and $p(x, -)p(-, y)$ to be close in all cases, making the CD small. But if xy is a common digraph then $p(x, y)$ will be significantly greater than $p(x, -)p(-, y)$, and if xy is an uncommon digraph then $p(x, y)$ will be significantly less than $p(x, -)p(-, y)$, and in either of these cases $(p(x, y) - p(x, -)p(-, y))^2$ will be relatively large.

Start MAGMA, and type `load "tut5data.txt";`

1. The file you have just loaded contains a sample piece of plaintext, called `sample`. To see the first 500 letters of it, type `sample[1..500];`. Now type

```
xx:=Decimation(sample,[4,5],10);
```

so that `xx` is the sequence of digraphs comprised of the 4th and 5th letters in every 10 letter block of `sample`. Type `xx[1..5];` to see the first 5 of these, and check that they are right. And type `CoincidenceDiscriminant(xx);` to see a typical value for the CD of a sequence of digraphs from standard text.

2. Type

```
for i:=1 to 9 do
   CoincidenceDiscriminant(Decimation(sample,[i,i+1],10));
```

```
end for;
```

and observe that you get similar values in all cases. Check also that other block lengths give similar values (e.g. by replacing 9 and 10 above by 12 and 13). And check that

```
CoincidenceDiscriminant(Decimation(sample,[1,4],10));
```

gives a noticeably smaller value, as do other cases where non-adjacent pairs of letters are chosen.

Suppose a message M is enciphered with a block transposition cipher, and let C be the ciphertext. Suppose that i, j, m are integers satisfying $1 \leq i < j \leq m$, and let $D(i, j, m)$ be the (i, j)-decimation of C of period m. If m happens to equal the block length of the cipher then there will be certain values of i and j for which $D(i, j, m)$ consists of pairs of letters that were adjacent in M. When this happens the CI and CD of $D(i, j, m)$ should approximate values typical of English text.

3. Type `ct[1..70];` to see the first 70 letters of the ciphertext we are going to tackle. Then type `SortedFreqDist(ct);` to check that it could well be produced by a transposition cipher. We are now going to use two procedures `CheckPeriod` and `FindAdjacencies` that have been defined in the MATH2068 startup file MagmaProcedures.txt. Type

```
CheckPeriod(ct,[2..15],0.007,0.005);
```

This causes MAGMA to check all the numbers m from 2 to 15 as possible periods. For each m it works out `CoincidenceIndex(Decimation(ct,[i,j],m))`, for values of i and j less than m, and tests whether the result exceeds 0.007. If so it tests if the coincidence discriminant exceeds 0.005. If this condition is also met (for some i and j) then it prints out "m is possible". (If you choose smaller numbers than 0.007 and 0.005 more "possibilities" will be found, higher numbers will give fewer. Multiples of the true period will appear as possibilities.)

None of the numbers from 2 to 15 were deemed to be possibilities; so now try

```
CheckPeriod(ct,[16..25],0.007,0.005);
```

You should find that 23 is possible. Then check 26 to 30 in the same way.

4. Having decided that 23 is the period, we now use the command

```
FindAdjacencies(ct,23,0.006,0.005);
```

This causes MAGMA to compute `CoincidenceIndex(Decimation(ct,[i,j],23))` and `CoincidenceDiscriminant(Decimation(ct,[i,j],23))` for all pairs i, j in the range from 1 to 23, looking for pairs such that the coincidence index is at least 0.006 and the coincidence discriminant is at least 0.005. It prints them all out. With any luck, for each i you will find two values of j that "want" to be moved next to i. Actually, if you are exceptionally lucky there will be exactly two values of i for which there is only one j that wants to be next to i – because the first and last letters in each block of 23 only have one neighbour in the block.

Anyway, you should be able to construct a sequence consisting of the numbers from 1 to 23 in some order, so that the numbers that most want to go next to each other are next to each other. Note that with this analysis there is no way to tell the correct sequence from its reverse; so you might wind up with text in which every block of 23 letters reads backwards. Anyway, type `ZZ:=TranspositionCryptosystem(23);` followed by a command **SIMILAR TO**

```
dk:=[19,16,23,13,7,5,2,9,8,20,1,11,14,10,4,3,12,21,15,18,6,22,17];
```

but USING THE SEQUENCE YOU FOUND in place of the one above. Then type

```
Enciphering(ZZ!dk,Encoding(ZZ,ct));
```

and see if you get anything readable. If not, replace `ZZ!dk` by `ZZ!Reverse(dk)`.

5. There is another test we could have used in place of the CI and CD. We could compute the quantity $\sum_{x,y}(p(x,y) - s(x,y))^2$, where $p(x,y)$ is the relative frequency of xy in the decimation we are investigating and $s(x,y)$ is its relative frequency in a piece of normal text, like `sample`. The startup file MagmaProcedures.txt defines a function called `CompareDigraphs` that does this. To test this, try

```
CompareDigraphs(Decimation(ct,[i,j],23),sample)
```

for various `i` and `j`, and check that it is much lower when `j` follows `i` in the key than in other cases.

6. Let $r_0 = 4$ and $r_{i+1} = r_i^2 - 2$. The *Lucas–Lehmer test* says that if p is an odd prime then the Mersenne number $m = 2^p - 1$ is prime if and only if m is a divisor of r_{p-2}. Type

```
CheckMersenne:=procedure(p)
 m:=2^p-1;
 r:=4;
 for i:=1 to p-2 do
  r:= (r^2-2) mod m;
 end for;
 if r eq 0 then print "M(",p,")=",m,"is prime";
 else print "M(",p,")=",m,"is not prime";
 end if;
end procedure;
```

thereby defining a procedure that implements the Lucas–Lehmer test, and then test it with the commands `CheckMersenne(3);` `CheckMersenne(31);` `CheckMersenne(107);` `CheckMersenne(4421);` and `CheckMersenne(4423);`.

(In fact MAGMA has unbelievably fast general methods for testing primality. To see this, try the commands `IsPrime(2^500+135);` `IsPrime(2^4423-1);` for example.)

***7.** Find a prime p such that $2p-1$ is prime and $\phi(4p-2) = \phi(4p-1) = \phi(4p)$. (You can use a `repeat` ... `until` loop, and Magma's functions `IsPrime`, `NextPrime` and `EulerPhi`.)