Computer Science

Dr. Y. Liow (January 29, 2021)

Contents

202 Classical cryptography **290500**
202.1 Classical Cryptosystems 290502
202.2 Caesar cipher 290503
202.3 Encryption and Decryption 290506
202.4 Numbers and letters 290509
202.5 Modular arithmetic 290510
202.6 Attacks 290551
202.7 Attacking the Shift Cipher 290553
202.8 Affine Cipher 290559
202.9 Attacking the Affine Cipher 290563
202.10 Vigenere Cipher 290565
202.11 Attacking the Vigenere Cipher 290567
202.12 Substitution Cipher 290579
202.13 Block and Stream Ciphers 290580
202.14 Hill's cipher 290581
202.15 Permutation cipher 290590
202.16 One time pad 290609
202.17 Linear Feedback Shift Register 290611

Chapter 202

Classical cryptography

File: chap.tex

```
File: classical-crypto.tex
```

202.1 Classical Cryptosystems

The subtitle: Stuff that you should not use anymore.

However some of these old stuff is important because their ideas are used in modern-day cryptography.

Here we go ...

File: caesar.tex

202.2 Caesar cipher

This is one of the earliest cryptosystems and apparently Julius Caesar used it (high tech, eh?). To encrypt a message, you simply do the following:

$$\alpha \to \delta, \ldots$$

or in our alphabet system:

$$a \to d, \;\; b \to e, \;\; c \to f, \;\; \ldots$$

i.e. a is replaced by d, b is replaced by e, etc. And of course you "go around in a circle": x is replaced by a, y is replaced by b, z is replaced by c.

This is known as the **Caesar cipher**. Caesar cipher

Here's a simple Python code to encrypt a character:

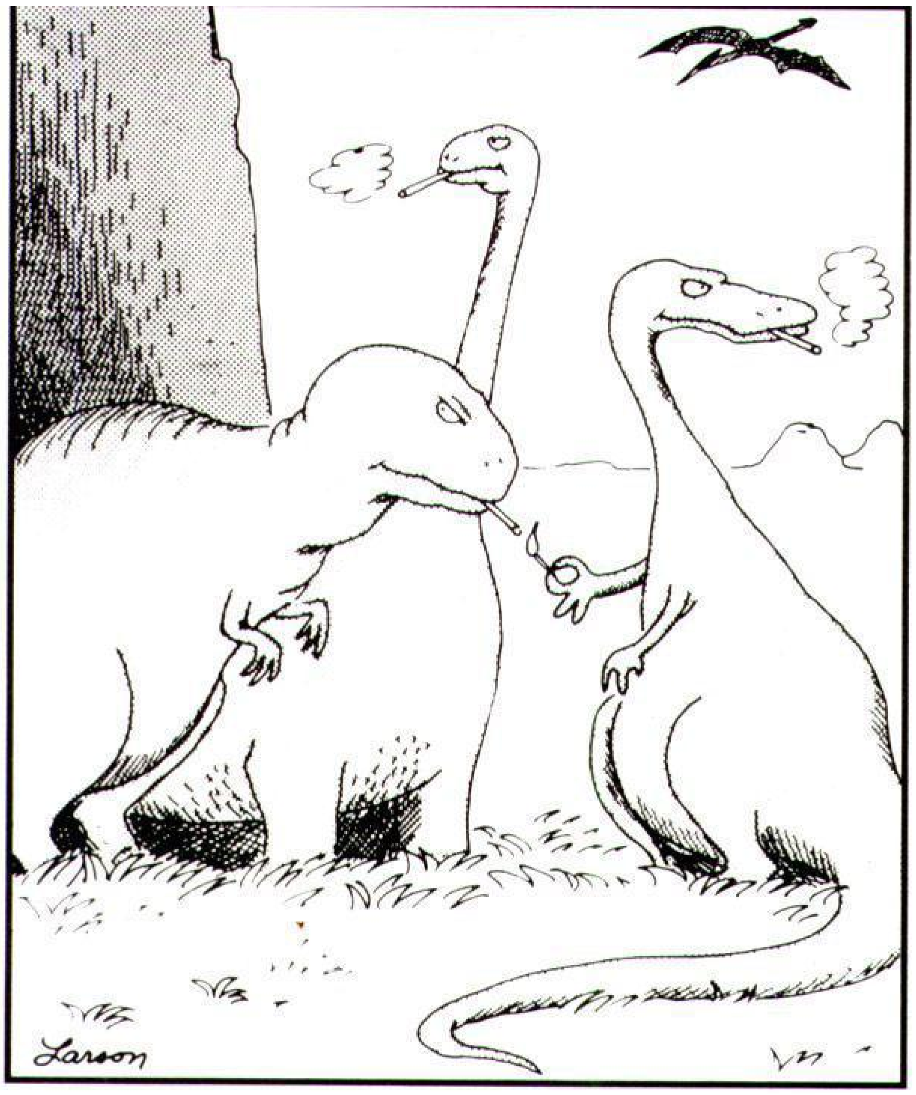
```
def E(x):
    i = ord(x) - ord('a')
    i = (i + 3) % 26
    return chr(ord('a') + i)
```

and for C++:

```
char E(char x)
{
    return (x - 'a' + 3) % 26 + 'a';
}
```

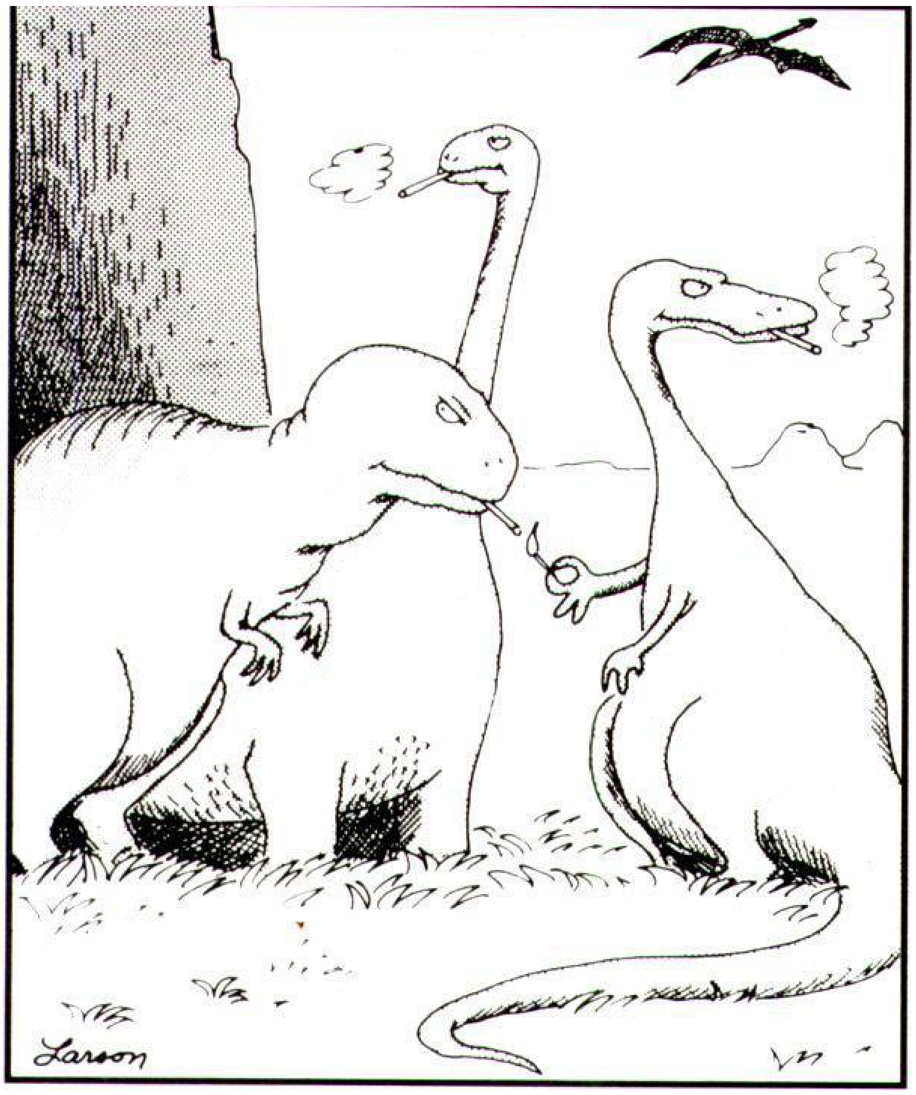
Of course there's no reason why you must "shift by 3". you can also shift 7. Right? Note that the encryption is only a single character. Of course for a string, you just encrypt character-by-character. Also, uppercase is replaced by lowercase. Furthermore, anything not a-z (example: punctuations, spaces) is thrown away.

Easy, right? OK. Break the following the code:



wkhuhdouhdvrqglqrvdxuvehfdphhawlqfw

*** WARNING: SPOILERS ON THE NEXT PAGE ***



The real reason dinosaurs became extinct

Exercise 202.2.1. Do you think Caesar cipher is secure? Do you think Julius Caesar was smart?

202.3 Encryption and Decryption

Given P and C be two sets. A **cipher** is just a pair of functions $E : P \to C$ and $D : C \to P$ where E is called the **encryption** and D is called the **decryption** such that

cipher

encryption
encryption

$$D(E(x)) = x \quad \text{for all } x \in P$$

In other words if you decrypt what you have encrypted, you get back the same data. (It'd better be so!) An element of P is called a **plaintext** – it's what you encrypt. An element of C is called a **ciphertext** – it's what you get when you encrypt.

plaintext

ciphertext

Caesar cipher is an example of a cipher. For Caesar cipher $P = C = \{a, b, c, ..., z\}$. Also, although the encryption function maps one character to another, it's understood that if you want to encrypt a string, you simply encrypt each character of the string and join them up into a string.

But if you allow the shift amount in the Caesar cipher to change, then the encryption and decryption depends on the shift amount – the key. A general principle in cryptography is the following:

Kerckhoffs' principle (1883): A secure cipher should not depend on the secrecy of the encryption and decryption algorithm, but rather on the secrecy of the key.

Kerckhoffs's principle

(Auguste Kerckhoffs: https://en.wikipedia.org/wiki/Auguste_Kerckhoffs)

The opposite and a really bad idea is called **security through obscurity**: , i.e., it's the hope that your encrypted messages are safe as long as the encryption and decryption algorithm are kept secret.

security through obscurity

In fact in modern cryptography, once a cipher is designed, the cryptography researcher(s) is expected to publish the cipher so that other researchers can check if the cipher is actually secure.

Why is this important? Because it's easy to change the key while changing the encryption and decryption algorithm might not be that easy. If a worker who performs the encryption or decryption process is captured, then he/she can be made (tortured?) to reveal the algorithm. On the other hand, if the key is stolen, then we can simply change the key. So in cryptography, it is always assumed that the algorithms (the cipher) cannot be kept secret for long.

So we just need to modify the definition of our cipher:

Let P be a set of plaintexts, C be a set of ciphertexts, and K be a set of keys. A **cipher** is a pair of functions $E : K \times P \to C$ and $D : K \times C \to P$ such that if $k \in K$,

cipher

$$D(k, E(k, x)) = x \quad \text{for all } x \in P$$

Notice that in the above the key used for encryption k is the same as the key used for decryption. To be more precise the above is a **symmetric cipher** . A symmetric cipher is also called a **private key cipher** because the key must be kept secret. Instead of writing $E(k, x)$ and $D(k, x)$, it's also common to write $E_k(x)$ and $D_k(x)$. Depending on which book you read, the encryption and decryption functions can also be written e instead of E and d instead of D.

symmetric cipher

private key cipher

An **asymmetric cipher** also called a **public key cipher** is a cipher where there are two distinct keys, one for encryption and one for decryption. It's called a public key cipher because the encryption key can be made public while the decryption key must be kept secret. In this case, if k, k' are the encryption and decryption keys, then the cipher must satisfy

asymmetric cipher
public key cipher

$$D(k', E(k, x)) = x$$

for $x \in P$. We won't see public key ciphers for while.

You can think of the Caesar cipher as a cipher that uses the key 3:

- encryption is "shift forward by 3"
- decryption is "shift backward by 3".

Generalizing the Caesar cipher, we get the shift cipher:

- encryption is "shift forward by k"
- decryption is "shift backward by k".

where k is the key. The shift cipher with key of 3 is just Caesar's cipher. I hope it's clear that the shift cipher with key 27 is the same as the shift cipher with key 1. Effectively speaking there are only 26 shifts, including the very bad key of 0. Hence for the Caesar cipher, $K = \{0, 1, 2, ..., 25\}$.

Exercise 202.3.1. Is the Caesar's cipher symmetric or asymmetric? (Duh)
☐

For classical ciphers, assuming we're only interested in English, the plaintexts are strings involves a-z. I will write $\{a, b, c, ..., z\}^*$ for the set of all strings with characters from $\{a, b, c, ..., z\}$. If n is a positive integer, I will also write $\{a, b, c, ..., z\}^n$ for the set of strings with length n and with characters from the set $\{a, b, c, ..., z\}$. For instance

$$\{a, b, c, ..., z\}^2 = \{aa, ab, ac, ..., zx, zy, zz\}$$

Exercise 202.3.2. How many strings are there in $\{a, b, c, ..., z\}^2$? How many strings are there in $\{a, b, c, ..., z\}^n$? □

In modern day cryptography, we frequently work with bit strings. In that case the set of all bit strings would be denoted by $\{0, 1\}^*$. Bit strings of length exactly 8 would be denoted by $\{0, 1\}^8$ – these would be bytes. For instance you might have heard of the SHA2 family of hash function. SHA256 takes in bit strings and spits out bit strings of length 256. So SHA256 is a function of type

$$\{0, 1\}^* \to \{0, 1\}^8$$

(Technically speaking SHA256 inputs do have a maximum limit in length, but it's so huge that for practical purposes it's as good as all possible bit strings.)

File: numbers-and-letters.tex

202.4 Numbers and letters

You know this is coming ... we'll be using *lots* of math to do encryption and decryption. In particular, for this notes, we associate letters a to z with numbers. The encryption and decryption function will work with either numbers of letters. Specifically we have the following correspondence:

In math, "correspondence" is the same as "1-1 correspondence" which is the same as "bijection". Remember bijection? It's time to check your discrete math notes

$$\begin{aligned} a &\leftrightarrow 0 \\ b &\leftrightarrow 1 \\ c &\leftrightarrow 2 \\ &\vdots \\ z &\leftrightarrow 25 \end{aligned}$$

if E encrypts a to c, I will say either

$$E(a) = c$$

or

$$E(0) = 2$$

Now you might say ... "so what's the big deal? Why rewrite a as 0, b as 1, etc. I can also come up with some secret encoding for instance why can't I rewrite a as a square, b as a triangle, etc.?"

Well ... the reason is because $0, 1, 2, \ldots$ are numbers ... and ...

they have operations (addition, subtraction, multiplication, division).

File: modular-arithmetic.tex

202.5 Modular arithmetic

Number theory is basically the study of whole numbers. That however is an over-simplification. The study of number theory involves almost all areas of mathematics. In fact many areas of mathematics were created just to study certain problems in number theory.

Note that number theory is an extremely big area in Mathematics and Computer Science. It is also extremely fascinating. Many problems is number theory can be stated very simply so that even a high school student can understand the statement of the problem. And yet the techniques used to solve some of these problems require the mathematical tools from almost every area of Math. Gauss once said that "Mathematics is the queen of sciences and number theory is the queen of mathematics".

Although there are many branches within Number Theory, right now we only need to know a little bit about Elementary Number Theory. "Elementary" here does not mean simple (although it will be for us since you're only seeing a small part of Elementary Number Theory). It means we are studying Number Theory using only properties of whole numbers (integers). Research in Number Theory requires real numbers, complex numbers, calculus, geometry, complex analysis, etc.

This will be a very short introduction to the vast area of Number Theory. In fact this is only a tiny fraction of Elementary Number Theory. This is one of the oldest area of Mathematics and one of the fascinating because of its history. If you want to learn more about number theory, just let me know. I can easily find a project for you to work on.

For now, we will look at modular arithmetic. Besides cryptography, modularity arithmetic is also used in data compression and error correction codes.

The set of integers $\{..., -3, -2, -1, 0, 1, 2, 3, ...\}$ denoted $\mathbb{Z}$ has two operations $+$ and $\cdot$. In terms of the algebraic structure (i.e. the operations), $\mathbb{Z}$ is known as a **commutative ring**. Basically a commutative ring is a set of "things" with two operations, addition and multiplication, which with rules that look like the addition and multiplication rules for $\mathbb{Z}$. For instance one such rule in $\mathbb{Z}$ is

$$a(b + c) = ab + ac$$

This same rule holds true for $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ and polynomails with coefficients in $\mathbb{Z}$.

The reason why mathematicians even bother defining this concept of "commutative ring" is more or less the same reason why we write functions in programs: for re-use. There are *many* naturally occurring rings. So if while developing the theory for $\langle \text{blah}_1 \rangle$ and $\langle \text{blah}_2 \rangle$ and they are both rings, then it's enough to prove a general fact that applies to both and quote the fact. This is also related to the concept of inheritance and abstract base classes. You can think of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ as subclasses of `CommutativeRing`. Therefore if you have a function

```
void f(CommutativeRing & r)
{
   ...
}
```

then `f` can work with `x` if `x` is a object of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$.

The mathematician develop general results while the programmer write general functions working on abstract base class objects. The idea is the same. The reason for generality is efficiency.

Here's a very important advice on studying rings, groups, fields, math, etc. (in fact this applies to any area of study where there is a great deal of generalization): Although the definition and theorems are general, you *always* keep a couple of standard examples in your mind as you read the statements. While reading them, mentally substitute your examples into the facts so that you can associate it to something more concrete. This is not just a learn technique for undergraduate students. Even researchers do that when they read research papers. For the definition of ring below, think of the ring of integers.

I will try to be as informal as possible for now so that you can develop some feel/intuition for what we need for now. Later I'm going to come back to this topic and redo the whole thing rigorously. The focus for now is to understand modulo 26 arithmetic.

Anyway, a **commutatative ring** R is a set of "things" with two operations abstractly denoted by $\oplus$ and $\odot$ ("addition" and "multiplication") Furthermore there are two special "things" in R which we will call 0_R and 1_R. The properties satisfied by $R, 0_R, 1_R, \oplus, \odot$ (of course there must be something satisfied by them!) are as follows. For $\oplus$ the properties are:

Example: the integer $\mathbb{Z}$,...+ and $\cdot$ of $\mathbb{Z}$ ...

0 and 1 of $\mathbb{Z}$ ...

1. $r \oplus s$ is in R for all r, s in R

   $i + j$ is in $\mathbb{Z}$ for integers i, j, ...

2. $(r \oplus s) \oplus t = r \oplus (s \oplus t)$ for all r, s, t in R

   $(i+j)+k = i+(j+k)$ for integers i, j, k ...

3. $r \oplus 0_R = r = 0_R \oplus r$ for all r in R

$i + 0 = i = 0 + i$ for integer i ...

4. For all r in R there is something in R which we will call r' such that $r \oplus r' = 0_R = r' \oplus r$

If i is an integer, then $-i$ is also an integer and $i + (-i) = 0 = (-i) + i$.

For $\odot$ the properties are:

1. $r \odot s$ is in R for all r, s in R

$ij = ji$ for integer i, j,...

2. $(r \odot s) \odot t = r \odot (s \odot t)$ for all r, s, t in R

$(ij)k = i(jk)$ for integers i, j, k

3. $r \odot 1_R = r = 1_R \odot r$ for all r in R

$i1 = i = 1i$ for integer i ...

4. $r \odot s = s \odot r$ for all r, s in R

$ij = ji$ for integers i, j

The property involving both $\oplus$ and $\odot$ is

1. $r \odot (s \oplus t) = r \odot s \oplus r \odot t$

$i(j + k) = ij + ik$ for integers i, j, k. Phew! So $\mathbb{Z}$ is a commutative ring.

Just remember this: A ring is a set of things with addition and multiplication. And when you're lost just think of the set of integers and its operations.

Now we'll be working with the alphabet a,b,...,z. We'll call them by their new identities: $0, 1, ..., 25$. This is not exactly all of $\mathbb{Z}$. The formulas for the encryption and decryption of Caesar's cipher involves addition and substraction. What if we go beyond? What is 3 + 25 (i.e., d + z)? No problem, we will take remainders mod 26. So instead of 3 + 25 we think of (3 + 25) mod 26 instead. Of course the remainders are 0, 1, ..., 25 which is exactly what we want.

In C++-speak, we take %26.

To indicate that we're only interested in remainders or more accurately, we ignore multiples of 26, we write

$$\begin{aligned} 3 + 25 &= 28 \\ &\equiv 2 \pmod{26} \end{aligned}$$

In general, if x and y are integers, we write

$$x \equiv y \pmod{26}$$

if 26 divides $x - y$. It does not mean that x is equal to y. It means that x and y are the same if you ignore additive multiples of 26, i.e.,

$$x \equiv y \pmod{26}$$

is the same as saying

$$x = y + (\text{... some multiple of 26 ...})$$

This is the same as saying the remainder when x is divided by 26 is the same as the remainder when y is divided by 26.

If two numbers differ by a multiple of 26, we say that they are **congruent** mod 26. congruent

Note that

$$\begin{aligned} 26 &\equiv 0 \pmod{26} \\ 27 &\equiv 1 \pmod{26} \\ 28 &\equiv 2 \pmod{26} \\ &\cdots \end{aligned}$$

and

$$\begin{aligned} -1 &\equiv 25 \pmod{26} \\ -2 &\equiv 24 \pmod{26} \\ -3 &\equiv 23 \pmod{26} \\ &\cdots \end{aligned}$$

So in the mod 26 world, since you are ignoring multiples of 26, *in some sense* there are only 26 numbers:

$$0, 1, 2, 3, ..., 25$$

I'll write $\mathbb{Z}/26$ for this world of 26 values. Remember that in this world, you can write the symbol

$$28$$

but this is the same as 2 in $\mathbb{Z}/26$:

$$28 \equiv 2 \pmod{26}$$

Of course in $\mathbb{Z}$, these symbols, i.e. 28 and 2, are different.

Most of the algebraic rules involving $+, -, *, 0, 1$ applies when working with integers mod 26. For instance suppose

$$x \equiv y \pmod{26}$$

where x and y are integers (i.e., x differs from y by a multiple of 26), then

$$x + z \equiv y + z \pmod{26}$$

where z is an integer. Likewise from

$$x \equiv y \pmod{26}$$

we get

$$xz \equiv yz \pmod{26}$$

It's also true that

$$x + 0 \equiv x \pmod{26}$$

and

$$1 \cdot x \equiv x \pmod{26}$$

To be more precise, $\mathbb{Z}/26$ is a commutative ring. It's a finite commutative ring with 26 values. Let me rewrite the axioms for a commutative ring for $\mathbb{Z}/26$.

For $+$ on $\mathbb{Z}/26$, the properties are:

1. $(r + s) \pmod{26}$ is in $\mathbb{Z}/26$ for all r, s in $\mathbb{Z}/26$
2. $(r + s) + t \equiv r + (s + t) \pmod{26}$ for all r, s, t in $\mathbb{Z}/26$
3. $r + 0 \equiv r \equiv 0 + r \pmod{26}$ for all r in $\mathbb{Z}/26$. In fact r' is just $(26 - r) \pmod{26}$. For instance for $r = 2 \pmod{26}$, $r' = 26 - 2 = 24 \pmod{26}$.
4. For all r in $\mathbb{Z}/26$ there is something in $\mathbb{Z}/26$ which we will call r' such that $r + r' \equiv 0 \equiv r' \oplus r \pmod{26}$

For $\cdot$ the properties are:

1. $r \cdot s \pmod{26}$ is in $\mathbb{Z}/26$ for all r, s in $\mathbb{Z}/26$
2. $(r \cdot s) \cdot t \equiv r \cdot (s \cdot t) \pmod{26}$ for all r, s, t in $\mathbb{Z}/26$
3. $r \cdot 1 \equiv r \equiv 1 \cdot r \pmod{26}$ for all r in $\mathbb{Z}/26$
4. $r \cdot s \equiv s \cdot r \pmod{26}$ for all r, s in $\mathbb{Z}/26$

The property involving both $+$ and $\cdot$ is

1. $r \cdot (s + t) \equiv r \cdot s + r \cdot t \pmod{26}$

It's also not too surprising that you can talk about mod n for any positive integer n.

If I don't say so, when I want you write some $x \pmod{26}$, I mean the x such that $0 \leq x < 26$. Of course there's no difference in mod 26 between 2 and 28. But in mod 26, the values in $[0, 26)$ is the preferred range. Also, when I say simplify $27 \pmod{26}$, I mean $1 \pmod{26}$.

Note that since $\mathbb{Z}/26$ is *finite*, you can always solve any equation in mod 26. This is similar to boolean values: there are only two. To solve a boolean equation, you just try all possible boolean values. So to solve a $\mathbb{Z}/26$ equation, you just try all the 26 possible values on all the variables that appear in the equation.

Exercise 202.5.1.

1. True or false: $100 \equiv 74 \pmod{26}$
2. True or false: $-3 \equiv 133 \pmod{26}$
3. True or false: $-20 \equiv 21 \pmod{26}$
4. True or false: $7 \equiv 3 \pmod{3}$
5. True or false: $17 \equiv -3 \pmod{5}$
6. True or false: $21 \equiv 11 \pmod{8}$
7. True or false: $42 \equiv 0 \pmod{7}$

Exercise 202.5.2.

1. Simplify $100 \pmod{26}$.
2. If you have very simple calculator with $+, -, *, /$, how would simplify $131246845 \pmod{26}$? You have 10 seconds ... the clock is ticking ...
3. Simplify $3^3 \pmod{26}$.
4. Solve $2x + 1 \equiv 0 \pmod{26}$ by brute force. Use Python or C++.
5. Solve $10x + 20 \equiv 4x + 78 \pmod{26}$ by brute force. Use Python or C++.
6. Simplify $10x + 20 \equiv 4x + 78 \pmod{26}$ first, and then solve it by brute force. Do you get the same results as in the previous part?
7. Solve $42x^5 + 10x + 1 \equiv 73 \pmod{3}$.
8. Solve $(1000000x + 2)^3 \equiv 2 \pmod{3}$.

Exercise 202.5.3.

1. If $x \equiv 1 \pmod 7$ and $x \equiv 5 \pmod{13}$, what can you tell me about x?
2. If $x \equiv 1 \pmod 7$ and $x \equiv 5 \pmod{13}$, what can you tell me about $x \pmod{7 \cdot 13}$?
3. $x \equiv 2 \pmod{7{\cdot}13}$ what can you tell me about $x \pmod 7$ and $x \pmod{13}$?
4. True or false: $x^3 \equiv x^0 \pmod 3$ since the 3 in the exponent can be replaced by 0.
5. In $\mathbb{Z}$, is it true that $(x+y)^2 = x^2 + y^2$? Try some values for x and y.
6. In $\mathbb{Z}/2$, is it true that $(x+y)^2 = x^2 + y^2 \pmod 2$? Try all values for x and y.
7. In $\mathbb{Z}$, is it true that $(x+y)^3 = x^3 + y^3$? Try some values for x and y.
8. In $\mathbb{Z}/3$, is it true that $(x+y)^3 \equiv x^3 + y^3 \pmod 3$? Try all values for x and y.
9. In $\mathbb{Z}/n$, is it true that $(x+y)^n \equiv x^n + y^n \pmod n$? Can you prove your claim.

Exercise 202.5.4. Just like for boolean values, you can write down the complete behavior of the boolean and and boolean or and boolean not operators (these are called truth tables), you can also complete specify the complete behavior of addition in mod 26, "negative of" in mod 26, multiplication in mod 26, and also multiplicative inverse mod 26. The multiplicative inverse of x in mod 26 is just the number y mod 26 such that

$$xy \equiv 1 \pmod{26}$$

The multiplicative inverse of $x \pmod{26}$ is written $x^{-1} \pmod{26}$ – this is an integer mod 26!!! It's not a fraction in $\mathbb{R}$!!! Sometimes $x^{-1} \pmod{26}$ might not exist. In that case write None. Write down these 4 tables.

SOLUTION.

Addition table for $\mathbb{Z}/26$:

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0																										
1																										
2																										
3																										
4																										
5																										
6																										
7																										
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16																										
17																										
18																										
19																										
20																										
21																										
22																										
23																										
24																										
25																										

In the above, when I write 5, I meant of course $5 \pmod{26}$.

Multiplication table for $\mathbb{Z}/26$:

×	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0																										
1																										
2																										
3																										
4																										
5																										
6																										
7																										
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16																										
17																										
18																										
19																										
20																										
21																										
22																										
23																										
24																										
25																										

Negative of table for $\mathbb{Z}/26$:

$x \pmod{26}$	$-x \pmod{26}$
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	

Multiplicative inverse table for $\mathbb{Z}/26$:

$x \pmod{26}$	$x^{-1} \pmod{26}$
0	None
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	

It should be clear that $0 \pmod{26}$ does not have an inverse.

Exercise 202.5.5. Solve

$$3x = 8$$

in $\mathbb{Z}$. Can you? Now solve

$$3x \equiv 8 \pmod{26}$$

$\square$

Exercise 202.5.6.

1. Solve
$$3x = 1$$
in $\mathbb{Z}$. Of course you can't! Now solve
$$3x = 1$$
in $\mathbb{Q}$. Of course you can! We would say that $\frac{1}{3}$ is the multiplicative inverse of 3 in $\mathbb{Q}$. Also, we would say that, in $\mathbb{Z}$, 3 is not invertible. What about
$$3x \equiv 1 \pmod{26}$$
Can you? If you can then you have found a multiplicative inverse of 3 (mod 26).

2. Now try to solve
$$ax \equiv 1 \pmod{26}$$
for $a \equiv 0, 1, 2, ..., 25 \pmod{26}$. Which a's have multiplicative inverses? Is there a pattern to a's with multiplicative inverses mod 26 and who which do not?

3. Now try
$$ax \equiv 1 \pmod{N}$$
where N is a positive integer and $a \equiv 0, 1, 2, ..., N-1 \pmod{N}$; say you try $N = 4, 5, 6, 7, 8$. Notice something? See a pattern? Is there a pattern to a's with multiplicative inverses mod 26 and who which do not?

Exercise 202.5.7. Solve

$$3x = 13$$

in $\mathbb{Z}$. Of course you can't! Now solve

$$3x = 13$$

in $\mathbb{Q}$. Of course you can! Do you realize that you solved it using the multiplicative inverse of 3 in $\mathbb{Q}$? What about

$$3x \equiv 13 \pmod{26}$$

Are you going to use brute force and try all values (mod 26)? [HINT: You had better not.]

Exercise 202.5.8. Solve

$$3x - 5 \equiv 7x + 20 \pmod{26}$$

by brute force. Next, simplify the above first before solving it. Do you get the same solutions?

Exercise 202.5.9. Solve

$$3x + 12 \equiv 10x + 23 \pmod{26}$$

by brute force. Next, simplify the above first before solving it. [Are you sure it's really simplifed? HINT: Multiplificative inverse.] Do you get the same solutions?

Exercise 202.5.10.

1. Can you solve
$$x^2 \equiv 1 \pmod{26}$$
Of course one solution is 1 (mod 26). Can you find the other one in 1 second? Are there just two? Or are there more than two solution? These are (of course) square roots of 1, but in mod 26 and not $\mathbb{R}$.

2. What about
$$x^2 \equiv 2 \pmod{26}$$

3. Keep going ... try to solve
$$x^2 \equiv a \pmod{26}$$
for all cases. Draw a table for the square roots of $a \pmod{26}$ for all a's.

Exercise 202.5.11. Continuing the previous exercise ...

1. Next, try to solve
$$x^2 \equiv a \pmod{N}$$
for all $a \pmod{N}$ for at least 3 values of N. See any patterns?

2. The above is probably too tough. What if you try
$$x^2 \equiv 2 \pmod{p}$$
for primes p? Try a few primes (maybe 20-30) and make a table. Notice a pattern?

Exercise 202.5.12.

1. What are all the possible values of $x^2 \pmod 4$?

   [Note: The key thing to note is that $x \pmod 4$ can take all the values 0, 1, 2, 3 mod 4. But the squares $x^2 \pmod 4$ can only take values 0, 1 mod 4.]

2. What are all the possible values of $x^2 \pmod 5$? [Note: Like the above, notice that squares in mod 5 do not cover all the values 0, 1, 2, 3, 4.]

3. What about squares mod 7?

4. Prove that there are no integer solutions to

   $$5x^2 - 8y^2 = 3$$

   [HINT: Take mod 4 ... and ... Checkmate!] Without modular arithmetic, there's no clear way to solve this problem! Go number theory!]

Exercise 202.5.13.

1. Solve
$$5x^2 + y^2 = 3$$
[HINT: You don't really need number theory for this one. Why? But if you want to, imitate the solution for the previous problem.]

2. Solve
$$11y^2 - 5x^2 = 3$$
[HINT: This is just a slight change from the previous problem. *But* now you need number theory. Try mod 4. If it does not work, try mod 5. Repeat.]

3. Solve
$$y^2 - 5x^2 = 2$$

4. What about this one:
$$x^2 - 5y^2 = 1$$

[ASIDE. Integer solutions to $x^2 - dy^2 = 1$ has been studied since at least 400BC. This equation appear the Cattle Problem of Archimedes:

> If thou art diligent and wise, O stranger, compute the number of cattle of the Sun, who once upon a time grazed on the fields of the Thrinacian isle of Sicily, divided into four herds of different colours, one milk white, another a glossy black, the third yellow and the last dappled. In each herd were bulls, mighty in number according to these proportions: Understand, stranger, that the white bulls were equal to a half and a third of the black together with the whole of the yellow, while the black were equal to the fourth part of the dappled and a fifth, together with, once more, the whole of the yellow. Observe further that the remaining bulls, the dappled, were equal to a sixth part of the white and a seventh, together with all the yellow. These were the proportions of the cows: The white were precisely equal to the third part and a fourth of the whole herd of the black; while the black were equal to the fourth part once more of the dappled and with it a fifth part, when all, including the bulls went to pasture together. Now the dappled in four parts8 were equal in number to a fifth part and a sixth of the yellow herd. Finally the yellow were in number equal to a sixth part and a seventh of the white herd. If thou canst accurately tell, O stranger, the number of cattle of the Sun, giving separately the number of well-fed bulls and again the number of females according to each colour, thou wouldst not be called unskilled or ignorant of numbers, but not yet shall thou be numbered among the wise. But come, understand also all these conditions regarding the cows of the Sun. When the white bulls mingled their number with the black, they stood firm, equal in depth and breadth, and the plains of Thrinacia, stretching far in all ways, were filled with their multitude. Again, when the yellow and the dappled bulls were gathered into one herd they stood in such a manner that their number, beginning from one, grew slowly greater till it completed a triangular figure, there being no bulls of other colours in their midst nor none of them lacking. If thou art able, O stranger, to find out all these things and gather them together in your mind, giving all the relations, thou shalt depart crowned with glory and knowing that thou hast been adjudged perfect in this species of wisdom.

If W, X, Y, Z represents the number of white, black, yellow, dappled bulls, you will get a systems of 7 linear equations, the first two being

$$W = (1/2 + 1/3)X + Z$$
$$X = (1/4 + 1/5)Y + Z$$

together with some constraints such as $W + X$ must be a square. After some manipulations, it can be shown that the equation to solve looks like

$$x^2 - 410286423278424y^2 = 1$$

What was Archimedes thinking? You are find information on the Archimedes Cattle Problem on the web.]

Exercise 202.5.14. Consider

$$5x^2 - 8y^2 = 3z^2$$

where we are only interested in finding integer solutions. Of course $(0, 0, 0)$ is a solution, but that's easy. (Right?) In general, an equation like

$$ax^p + by^q = cz^r$$

where $p, q, r > 0$ always has $(0, 0, 0)$ as a solution. $(0, 0, 0)$ is sometimes called the trivial solution. So we might as well assume $(x, y, z) \neq (0, 0, 0)$, i.e., assume they are not all zero. By the way the polynomial

$$5x^2 - 8y^2 - 3z^2$$

is a sum of terms where the number of variables appearing in each term is the same, i.e., 2. Such polynomial are is said to be homogeneous of degree 2.

How many solutions can you find? Is there any at all? Is there a finite number of solutions? If there are infinitely many solutions, can you write them down? If not all of them, maybe an infinite family of them. For instance it would be nice to say: "For any integer n, $x = n, y = 2n, z = n + 5$ is a solution". This would be a 1–parameter family of solutions.

1. What's the first thing you should do?

2. Prove that if x, y, z is a solution, then nx, ny, nz is also a solution for any integer n.

3. In case you can't see a solution right away, do the following. If (x, y, z) are solutions, show that x and z must be even. [HINT: mod 4.] This is helpful since we won't have to check the odd x or odd z caaes. That cuts down a brute force search down by 3/4.

4. Continuing the above: let $x = 2a$, $y = b$, and $z = 2c$, substitute, and you'll get a new equation in a, b, c.

5. The equation in a, b, c is easier than the one in x, y, z. Why? Because a solution in the equation in a, b, c for the above question would correspond to a solution x, y, z in the original equatison and a, b, c solution is smaller. Can you now find some solutions for the equation in a, b, c?

6. I won't go further. But for those who want to solve this probably complete, let me just say that once you have one one solution, you can

parametrize all solutions using two integer parameters.

Exercise 202.5.15. Find all the integer solutions to $3x^2 + 4y^2 = 5z^2$.

SOLUTION. Of course $(x, y, z) = (0, 0, 0)$ is a solution since, for instance, the polynomial is homogeneous. If any two of x, y, z are 0s, then the third must also be 0.

Now consider the case where exactly one of x, y, z is 0.

1. If $x = 0$, then $4y^2 = 5z^2$. We can assume that all common factors between y, z are removed so that $\gcd(y, z) = 1$. (Suppose $g = \gcd(y, z)$. Let $y' = y/g$ and $z' = z/g$, We again would arrive at $4y'^2 = 5z'^2$.) Since $5 \mid 5z^2$, we have $5 \mid 4y^2$, and hence $5 \mid y$. Therefore $y = 5y'$. Hence $4(5y')^2 = 5z^2$, i.e., $4(5)y'^2 = z^2$, which implies that $5 \mid z$. This contradicts $\gcd(y, z) = 1$.

2. If $y = 0$, then $3x^2 = 5z^2$. We can assume that all common factors between x, z are removed so that $\gcd(x, z) = 1$. Since $3 \mid 3x^2$, we obtain $3 \mid 5z^2$, which implies that $3 \mid z$. Therefore $z = 3z'$ and hence $3x^2 = 5(3z')^2$. This implies that $x^2 = 5(3)z'^2$. Therefore $3 \mid x^2$, and hence $3 \mid x$. This contradicts $\gcd(x, z) = 1$.

3. If $z = 0$, then $3x^2 + 4y^2 = 0$. Since $3 \mid 3x^2$, we get $3 \mid 4y^2$, which implies that $3 \mid y$. Let $y = 3y'$. Then $3x^2 + 4(3y')^2 = 0$ and hence $x^2 = 4(3)y'^2$. Therefore $3 \mid x^2$ and hence $3 \mid x$. This contradicts $\gcd(x, y) = 1$.

Now suppose $(x, y, z) \neq (0, 0, 0)$.

Let x, y, z be a solution with $x > 0, y > 0, z > 0$. We may assume $\gcd(x, y, z) = 1$.

Method 1. Taking mod 3,

$$y^2 \equiv 2z^2 \pmod 3$$

Squares in mod 3 are 0 or 1 mod 3. If $z^2 \equiv 1 \pmod 3$, then $y^2 \equiv 2 \pmod 3$ which is impossible. Hence $z^2 \equiv 0 \pmod 3$ and therefore $y^2 \equiv 0 \pmod 3$. This implies that $3 \mid z^2$ and $3 \mid y^2$. Therefore Then

$$3 \mid y, \quad 3 \mid z$$

Let $y = 3y'$ and $z = 3z'$. Then

$$3x^2 + 4(3y')^2 = 5(3z')^2$$

i.e.,

$$x^2 + 12y'^2 = 15z'^2$$

which implies that $3 \mid x$. This is a contradiction since $\gcd(x, y, z) = 1$.

Method 2. Taking mod 4, we get

$$3x^2 \equiv z^2 \pmod 4$$

Then $2 \mid x$ and $2 \mid z$. Let $x = 2x'$ and $z = 2z'$. Then

$$3(2x')^2 + 4y^2 = 5(2z')^2$$

i.e.,

$$3x'^2 + y^2 = 5z'^2$$

Taking mod 4,

$$3x'^2 + y^2 \equiv z'^2 \pmod 4$$

Then

- Assume z' is even. Then either x', y are even or $x'^2 \equiv 1 \equiv y^2 \pmod 4$. If x', y are even and z' is also even, then $\gcd(x, y, z) \neq 1$ which is a contradiction. For the other case

  $$3(4m + 1)^2 + (4n + 1)^2 = (2z'')^2$$

  i.e.,

  $$3(16m^2 + 8m + 1) + (16n^2 + 8n + 1) = 4z''^2$$

  i.e.,

  $$12m^2 + 6m + 4n^2 + 2n + 1 = z''^2$$

  All we can say is z'' is odd. There's no clear path forward.
- Assume $z'^2 \equiv 1 \pmod 4$. Then $2|x'$ and $y'^2 \equiv 1 \pmod 4$.

There are more cases to consider and this seems to be a bad direction to take.

Method 3. Taking mod 5, we get

$$3x^2 + 4y^2 \equiv 0 \pmod 5$$

i.e.

$$3x^2 \equiv y^2 \pmod 5$$

Squares in mod 5 are $0, 1, 4 \pmod 5$. If $x^2 \equiv 1 \pmod 5$, then $y^2 \equiv 3x^2 \equiv 3 \pmod 3$ which is impossible. If $x^2 \equiv 4 \pmod 5$, then $y^2 \equiv 3x^2 \equiv 12 \equiv 2 \pmod 3$ which is again impossible. Hence $x^2 \equiv 0 \pmod 5$ and hence $y^2 \equiv 0$

(mod 5). Altogether we have

$$5 \mid x^2, \quad 5 \mod y^2$$

and therefore

$$5 \mid x, \quad 5 \mod y$$

Let $x = 5x'$ and $y = 5y'$. Then we have

$$3(5x')^2 + 4(5y')^2 = 5z^2$$

i.e.,

$$3(5)x'^2 + 4(5)y'^2 = z^2$$

which implies that $5 \mid z^2$. Therefore $5 \mid z$. This contradicts $\gcd(x, y, z) = 1$.
□

Notes.

- What about $9x^2 + 4y^2 = 5z^2$? How far can you go using the method above?
- What about $3x^2 + 4y^2 = 25z^2$? How far can you go using the method above?
- What about $9x^2 + 4y^2 = 25z^2$?
- What about $3x^2 + 2y^2 = 5z^2$?
- What about $3x^3 + 4y^3 = 5z^4$?

Exercise 202.5.16. Find all solutions to

$$y^2 = x^3 + x + 1 \pmod{26}$$

(No this is not a random, idle question. It's actually very important.)

Exercise 202.5.17. (This "idle" problem is way more important than you think.)

1. Compute
$$2^n \pmod{26}$$
for $n = 0, 1, 2, 3, ...$ What do you notice? Can you tell me what is
$$2^{1000} \pmod{26}$$

2. Now try $3^n \pmod{26}$ for $n = 0, 1, 2, 3, ...$ Can you tell me what is $3^{1000} \pmod{26}$

3. Now try $4^n \pmod{26}$ for $n = 0, 1, 2, 3, ...$ Can you tell me what is $4^{1000} \pmod{26}$

4. Now try $5^n \pmod{26}$ for $n = 0, 1, 2, 3, ...$ Can you tell me what is $5^{1000} \pmod{26}$

5. Of course since $\mathbb{Z}/26$ is finite, you would expect $a^n \pmod{26}$ to be finite even though $n = 0, 1, 2, ...$ runs through an infinite set. Which a would give you most distinct value of $a^n \pmod{26}$ as you run through all values for n? And do you notice a pattern in the finite collection of all the $a^n \pmod{26}$?

Exercise 202.5.18. Continuing the previous question ...

What about $a^n \pmod{p}$ when p is a prime? How many values do you get as you run through $n = 0, 1, 2, ...$? First try it for $a = 2$: try about 20 to 50 primes p. Do you see a pattern? Once you have spotted the pattern, try $a = 3$, etc.

Exercise 202.5.19.

1. What is 10 (mod 3)? (when simplified).
2. What is 100 (mod 3)? (when simplified).
3. What is 1000 (mod 3)? (when simplified).
4. What is 5342 (mod 3)? (when simplified).
5. What is 534603142235187 (mod 3)? (when simplified).
6. What is the general fact here?

Exercise 202.5.20.

1. What is $10 \pmod 9$? (when simplified).
2. What is $100 \pmod 9$? (when simplified).
3. What is $1000 \pmod 9$? (when simplified).
4. What is $9142 \pmod 9$? (when simplified).
5. What is $96331420351887 \pmod 9$? (when simplified).
6. What is the general fact here?

Exercise 202.5.21.

1. What is $10 \pmod{11}$? (when simplified).
2. What is $100 \pmod{11}$? (when simplified).
3. What is $1000 \pmod{11}$? (when simplified).
4. What is $9142 \pmod{11}$? (when simplified).
5. What is $80432440556787 \pmod{11}$? (when simplified).
6. What is the general fact here?

Exercise 202.5.22. However certain "bizarre" things do happen. In $\mathbb{Z}$ (in fact in $\mathbb{Q}$ and $\mathbb{R}$ as well), you have the implication

$$xy = 0 \implies x = 0 \text{ or } y = 0$$

Is it true that

$$xy \equiv 0 \pmod{26} \implies x \equiv 0 \pmod{26} \text{ or } y \equiv 0 \pmod{26}$$

For each N, check when

$$xy \equiv 0 \pmod{N} \implies x \equiv 0 \pmod{N} \text{ or } y \equiv 0 \pmod{N}$$

Exercise 202.5.23. (Dr.Liow's magic formula) Suppose I want to simplify 23532 (mod 26). With a calculator or C++ or python, we see quickly that

$$23532 \equiv 2 \pmod{26}$$

I claim that you can use this magic formula: Suppose the digits of a 5–digit number is *edcba*. For instance $edcba = 23532$ where $e = 2, d = 3, c = 5, b = 3, a = 2$. Then

$$edcba \equiv ba + 2(-2c + 6d - 5e) \pmod{26}$$

For instance when $edcba = 23532$, we have

$$edcba \equiv 32 + 2(-2 \cdot 5 + 6 \cdot 3 - 5 \cdot 2) \pmod{26}$$

and

$$32 + 2(-2 \cdot 5 + 6 \cdot 3 - 5 \cdot 2) \equiv 6 + 2(-2) \equiv 2 \pmod{26}$$

which is easier to work with than the much larger 23532. So ... is

$$edcba \equiv ba + 2(-2c + 6d - 5e) \pmod{26}$$

really true for any 5-digit number *edcba*? Write a program to test all cases. Next, write a proof (that does not involve checking all cases like a program).

Exercise 202.5.24. Continuing the previous question:

1. I further claim that if the number is an 8–digit number $hgfedcba$, then

   $$hgfedcba \equiv ba + 2((-2c + 6d - 5e) + (2f - 6g + 5h)) \pmod{26}$$

   This is also the same as saying

   $$hgfedcba \equiv ba + 2((2(f - c) + 6(d - g) + 5(h - e)) \pmod{26}$$

   Is this true? Write a program to check. If it's true, prove it.

2. Can you conjecture a general formula? Can you prove it?

Exercise 202.5.25.

1. Solve the following

$$\begin{aligned} 9x + 5y &\equiv 3 \pmod{26} \\ 5x + 7y &\equiv 1 \pmod{26} \end{aligned}$$

   First solving by writing a program that performs a brute force search for solutions. Next, try to solve it algebraically by hand.

2. What about this one:

$$\begin{aligned} 3x - y &\equiv 2 \pmod{26} \\ 2x + 19y &\equiv 14 \pmod{26} \end{aligned}$$

3. And this one:

$$\begin{aligned} 9x + y &\equiv 2 \pmod{26} \\ 19x + 5y &\equiv 7 \pmod{26} \end{aligned}$$

4. Write a program that solves

$$\begin{aligned} ax + b &\equiv c \pmod{26} \\ dx + ey &\equiv f \pmod{26} \end{aligned}$$

   for a, b, c, d, e, f in $\mathbb{Z}/26$ by brute force search. Then write a program that randomly picks a, b, c, d, e, f in $\mathbb{Z}/26$ and ask you to solve it. Print all the cases where a, b, c, d, e, f provides a linear system of two equations or two unknowns where there is no solution. Do you notice a pattern in these degenerate cases?

Exercise 202.5.26. Notice that earlier on, we wrote down multiplicative inverses of elements in $\mathbb{Z}/26$. Some elements do not have inverses. Can you tell if there's something common among them?

1. Compute the multiplicative inverses of elements in $\mathbb{Z}/3$.
2. Compute the multiplicative inverses of elements in $\mathbb{Z}/5$.
3. Compute the multiplicative inverses of elements in $\mathbb{Z}/7$.
4. Compute the multiplicative inverses of elements in $\mathbb{Z}/11$.
5. Do you notice something special about the above cases? How many elements have multiplicative inverse?
6. Compute the multiplicative inverses of elements in $\mathbb{Z}/6$.
7. Compute the multiplicative inverses of elements in $\mathbb{Z}/10$.
8. Compute the multiplicative inverses of elements in $\mathbb{Z}/14$.
9. Compute the multiplicative inverses of elements in $\mathbb{Z}/15$.
10. In the above 4 cases is there something special about values which are invertible? Don't see the pattern? In the above, the modulus are all products of two distinct primes.

Exercise 202.5.27. Suppose you want to write a random number generator. Say the numbers are to be in the range [0,256]. (You can try a bigger range later – don't be too ambitious for now.) Of course use (reasonable) formula.

1. Go ahead and try this one:

$$h(n) = (2n + 5) \pmod{256}$$

   Starting with a seed value of 1, the next value is $h(1) = 7$. And the value after that is $h(7) = 19$. And the next is $h(19) = 43$. What are all the possible values you can obtain using this h?

2. Now of course you do want to have lots of values. For instance you might want to use this hash function to build a hashtable. Or a cryptographic hash (which we have no covered yet). But in any case, you hope that h covers *all* the values in $[0, 255]$. Was the function above a good function?

3. What if you use a different seed value? What if you start with 2? Or 3?

4. Try to find a function that generates as many values in $[0, 255]$ as possible. Can you find one that covers the whole range of $[0, 255]$?

File: attacking-classic-crypto.tex

202.6 Attacks

OK. Now it's time to attack some of the classical cryptosystems. Before we do that we want to know exactly what is it we want to achieve.

Recall that a cryptosystems is made up of the encryption and decryption function E and D.

Suppose Alice wants to send a message m to Bob. She encrypts the message to get $E_k(m)$ and delivers $E_k(m)$ to Bob. Bob decrypts $E_k(m)$ by applying D_k and get $D_k(E_k(m)) = m$.

Now let's think about the rogue agent Eve. What does she want?

The most common mode of attack assumes Eve receives a copy of $E_k(m)$ and she wants to derive m. It's even better if she can derive k because in that case she actually has D_k and hence can decrypt all future ciphertexts. (Don't forget we assume that the encryption and decryption algorithm is known. That means if Eve has k, she has D_k and in fact also E_k. Only the key(s) is secret). Not only that. If she has k, she also has E_k and therefore she can actually impersonate Alice!

The various assumptions of what Eve has are called **attack models** or **attack modes**. Here are some standard ones.

attack models

attack modes

1. The **known ciphertext attack** is an attack where the ciphertext of a plaintext is available to the attacker. In other words, this attack involves computing k from $E_k(m)$:

   known ciphertext attack

   $$E_k(m) \mapsto k$$

   This include the case when there is more than one ciphertext.

2. The **known plaintext attack** is an attack where the Eve has some messages and their ciphertext. So her goal is this:

   known plaintext attack

   $$m_1, m_2, ..., m_n, E_k(m_1), E_k(m_2), ..., E_k(m_n) \mapsto k$$

   This does happen. For instance the breaking of Germany's Enigma ciphers during WWII is based on this assumption.

3. The **chosen plaintext attack** is an attack where the Eve can actually encrypt messages/plaintexts that she chooses. Make sure you see that this is different from known plaintext. chosen plaintext attack

4. The **chosen ciphertext attack** is an attack where the Eve can actually encrypt messages/plaintexts that she chooses. Make sure you see that this is different from known plaintext. chosen ciphertext attack

   [x] In the 'chosen ciphertext' attack, the attacker chooses a portion of the decrypted ciphertext. He then compares the decrypted ciphertext with the plaintext and figures out the key.

   Earlier versions of RSA were subject to these types of attacks.

202.7 Attacking the Shift Cipher

BRUTE FORCE.

Recall that for the shift cipher

$$E_k(x) \equiv (x + k) \pmod{26}, \qquad D_k(x) \equiv (x - k) \pmod{26}$$

Since $0 \leq k \leq 25$, there are not many possible values for k. So you can try to apply *all* the possible decryptions D_0, D_1, D_2, ..., D_{25}. Assuming the message is something Eve can read, all that is required is that Eve has enough time to read 26 messages. Easy!

HEURISTIC USING PROBABILITY

But in fact Eve can do better. If the message is long enough, based on letter frequencies, heuristically, she can try to tabulate the frequencies of all the characters and then assume the most common occurring character is decrypted as `e` which is statistically the most commonly occurring letter in English. Of course knowing how to decrypt to `e` is sufficient for Eve to decrypt all the letters right?

Too bad if the message is in Russian.

The following is a table of probabilities for each letter used in English.

Letter	Probability
`e`	0.127
`t`	0.091
`a`	0.082
`o`	0.075
`i`	0.070
`n`	0.067
`s`	0.063
`h`	0.061
`r`	0.060
`d`	0.043
`l`	0.040
`c`	0.028
`u`	0.028
`m`	0.024
`w`	0.023
`f`	0.022
`g`	0.020
`y`	0.020
`p`	0.019
`b`	0.015
`v`	0.010
`k`	0.008
`j`	0.002
`x`	0.001
`q`	0.001
`z`	0.001

Of course these are probabilities. It does not mean that the second most frequently occurring letter *must* be `t`! I've divided up the probabilities into groups according to jumps in the values.

It is also useful to know that besides commonly occurring letters, which *pairs* of letters occurring frequently next to each other. These are called **digrams** (or 2–grams). For three, they are called **trigrams** (or 3–grams). The following is a table of commonly occurring digrams and trigrams listed in decreasing order of frequencies:

digrams

trigrams

n	n-grams (in decreasing order)
2	th he in er an re ed on es st en at to nt ha nd ou ea ng as or ti is et it ar te se hi of
3	the ing and her ere ent tha nth was eth for dth

Here are the frequencies of the 2-grams (of course not 2-grams, just the top few):

2-gram	Probability
th	0.0271
he	0.0233
in	0.0203
er	0.0178
an	0.0161
re	0.0141
es	0.0132
on	0.0132
st	0.0125
nt	0.0117
en	0.0113
at	0.0112
ed	0.0108
nd	0.0107
to	0.0107
or	0.0106
ea	0.0100
ti	0.0099
ar	0.0098
te	0.0098
ng	0.0089
al	0.0088
it	0.0088
as	0.0087
is	0.0086
ha	0.0083
et	0.0076
se	0.0073
ou	0.0072
of	0.0071

3-gram	Probability
the	0.0181
and	0.0073
ing	0.0072
ent	0.0042
ion	0.0042
her	0.0036
for	0.0034
tha	0.0033
nth	0.0033
int	0.0032
ere	0.0031
tio	0.0031
ter	0.0030
est	0.0028
ers	0.0028
ati	0.0026
hat	0.0026
ate	0.0025
all	0.0025
eth	0.0024
hes	0.0024
ver	0.0024
his	0.0024
oft	0.0022
ith	0.0021
fth	0.0021
sth	0.0021
oth	0.0021
res	0.0021
ont	0.0020

4-gram	Probability
`tion`	0.31
`nthe`	0.27
`ther`	0.24
`that`	0.21
`ofth`	0.19
`fthe`	0.19
`thes`	0.18
`with`	0.18
`inth`	0.17
`atio`	0.17
`othe`	0.16
`tthe`	0.16
`dthe`	0.15
`ingt`	0.15
`ethe`	0.15
`sand`	0.14
`sthe`	0.14
`here`	0.13
`thec`	0.13
`ment`	0.12
`them`	0.12
`rthe`	0.12
`thep`	0.11
`from`	0.10
`this`	0.10
`ting`	0.10
`thei`	0.10
`ngth`	0.10
`ions`	0.10
`andt`	0.10

So a slight improve to brute force search of trying $k = 0, 1, 2, \ldots, 25$, is to try encrypt `e` to the most commonly occurring letter, the second, the third, etc.

(Some authors use M for their set of plaintexts instead of P. In that case they might call their plaintexts messages instead.)

Exercise 202.7.1. Decrypt this given that this is encrypted using a shift cipher:

```
tozyevyzhhspcpxjtopldnzxpqczxthtwwloxtes
zhpgpceslezypvpjtyrcpotpyetdnlqqptyptrpe
lnzfawpnfadzqnzqqpptyezxplyohptcoestyrdu
fdedelceezslaapy
```

Exercise 202.7.2. Of course the method still requires a human being to read the decrypted text. How would you improve your program so that even this is automated? This is just for you to think about. If you can implement it, good. I just want to hear a strategy. (This is obvious if you have taken CISS358.)

`File: affine.tex`

202.8 Affine Cipher

Suppose we use $\mathbb{Z}/26$ instead of `'a'` to `'z'` again. Recall that the shift cipher is

$$E(k, x) = (x + k) \pmod{26}$$

and

$$D(k, x) = (x - k) \pmod{26}$$

The benefit of translating our encryption/decryption 'shift up" and 'shift down" into mathematical operations in $\mathbb{Z}/26$, is that we can now generalize and use different mathematical formulas!

Here's the affine cipher. The encryption algorithm for the affine cipher looks like this:

$$E((a, b), x) = (ax + b) \pmod{26}$$

Note that the key is not one single number – the key is (a, b) mod 26 which is made up of two numbers. Why is that important?

Because this means that there are more key values! Which means that Eve has to try more key!!!! Get it??

But what is the decryption function? Of course we know that

$$D((a, b), E((a, b), x)) = x \pmod{26}$$

This means that

$$D((a, b), ax + b) = x \pmod{26}$$

Suppose I make a guess ... When I look at the shift cipher, I notice that the decryption function is similar in form to the encryption function. Maybe the decryption function for the affine cipher is similar in form to the encryption function???

Let's try

$$D((a, b), x) = cx + d \pmod{26}$$

In that case, from

$$D((a, b), ax + b) \equiv x \pmod{26}$$

we would get

$$c(ax+b)) + d \equiv x \pmod{26}$$

Which gives us

$$cax + cb + d \equiv x \pmod{26}$$

Now what? Well maybe

$$cax \equiv x \pmod{26}$$

and

$$cb + d \equiv 0 \pmod{26}$$

The first condition

$$cax \equiv x \pmod{26}$$

is achieved is we have

$$ca \equiv 1 \pmod{26}$$

Can this be done?

Exercise 202.8.1. Suppose $a = 1$. What c's would make $ca \equiv 1 \pmod{26}$? What if $a = 2$? What about $a = 3$? Etc.

Given an integer a, if c satisfies

$$ca \equiv 1 \pmod{26}$$

we say that c is a **multiplicative inverse** of a mod 26. We usually write c as a^{-1} mod 26. Note that a^{-1} mod 26 is NOT a fraction!!! It's a whole number. If a has a multiplicative inverse mod 26, we say that a is **invertible** mod 26.

multiplicative inverse

invertible

Exercise 202.8.2. Draw a table of a^{-1} mod 26 for all a's in mod 26. If the multiplicative inverse is not defined, write UNDEFINED.

Exercise 202.8.3. How many integers $0, 1, 2, ..., 25$ have multiplicative inverses mod 26?

Therefore to satisfy

$$ca \equiv 1 \pmod{26}$$

we can't just pick any a. We have to pick an a with a multiplicative inverse mod 26.

After we have chosen a good a, what do we do? We then have

$$D((a,b),x) = cx + d \pmod{26}$$

where c is the multiplicative inverse of a mod 26. But what about d???

Remember we still have the condition

$$cb + d \equiv 0 \pmod{26}$$

Writing $a^{-1} \bmod 26$ for c, we get

$$a^{-1}b + d \equiv 0 \pmod{26}$$

we get

$$d \equiv -a^{-1}b \pmod{26}$$

Therefore we have the following: The affine cipher is

$$E((a,b),x) = (ax + b) \pmod{26}$$

where a is invertible mod 26 and

$$\begin{aligned} D((a,b),x) &\equiv a^{-1}x - a^{-1}b \pmod{26} \\ &\equiv a^{-1}(x - b) \pmod{26} \end{aligned}$$

Exercise 202.8.4. Write down the (simplified) encryption and decryption function for the affine cipher when the key is $(3, 12)$. Encrypt `gollum` and then decrypt to check that you get back `gollum`.

The above however uses a lot of "iffy" math. For instance we used the fact

$$a(b + c) \equiv ab + ac \pmod{26}$$

(where?) We also use the fact that if

$$a + b \equiv 0 \pmod{26}$$

then

$$a \equiv -b \pmod{26}$$

We seem to be treated math in mod 26 like math in $\mathbb{Z}$ and $\mathbb{R}$!!! Is that justifiable? It turns out that the above algebra is actually correct. I'll have to come back to that later otherwise people will think we are rambling nonsense and making things up.

Exercise 202.8.5. What is the size of the key space for the affine cipher? In the worse case how many tries must Eve attempt before breaking an affine cipher? Compare this with the shift cipher.

`File: attacking-affine.tex`

202.9 Attacking the Affine Cipher

Recall the encryption and decryption of the affine cipher looks like

$$E_{a,b}(x) \equiv (ax + b) \pmod{26}, \qquad D_{a,b}(x) \equiv a^{-1}(x - b) \pmod{26}$$

Again you can do a brute force search for a, b. After all there are not that many possibilities for a and b. But we can do better if we use letter (1–gram) frequencies again. Again suppose you have computed the frequencies of the letters of the ciphertext and say that `g` is the most common letter. So you assume `e` is encrypted as `g`. This is the same as saying 4 is encrypted as 6, i.e.,

$$E_{a,b}(4) = 6$$

right? Now using the formula for $E_{a,b}$ we get

$$4a + b = 6$$

To be accurate the equation should be

$$4a + b \equiv 6 \pmod{26}$$

Now suppose the second most common letter in the ciphertext is `y`. So you assume that `t` is encrypted as `y`. This means

$$20a + b \equiv 24 \pmod{26}$$

Right? Yes, no? Think about it. So you can solve for a and b from the linear equations

$$\begin{aligned} 4a + b &\equiv 6 \pmod{26} \\ 20a + b &\equiv 24 \pmod{26} \end{aligned}$$

Exercise 202.9.1. Solve the above for a and b.

Exercise 202.9.2. Now let's abstract the above. Suppose A, B, C, D are numbers and a and b satisfy the equations

$$\begin{aligned} Aa + b &= B \\ Ca + b &= D \end{aligned}$$

Solve for a and b in terms for A, B, C, D. If this can be done, then of course you can write a C++ function to return a and b directly.

Exercise 202.9.3. But wait! a is not arbitrary! Remember that a must be invertible in mod 26. Recall that this means there is some integer c such that $ac \equiv 1 \pmod{26}$. Therefore it would be helpful if you have a function that will determine if a is invertible mod 26. How would you do that?

File: vigenere.tex

202.10 Vigenere Cipher

The shift, affine, substitution ciphers are called **monoalphabetic** ciphers – each letter can be mapped to only one letter.

monoalphabetic

A **polyalphabetic** ciphers is the opposite of monoalphabetic ciphers.

polyalphabetic

The Vigenere cipher is basically a collection of shifts. Here's an example.

My key in going to be the word `fun`. Note that when I translate `fun` to numbers (with `a` $\rightarrow 0$, `b` $\rightarrow 1$, `c` $\rightarrow 2$, ...) I get $5, 20, 13$. So all I need to do is to encrypt the characters by doing

- shift by 5
- shift by 20
- shift by 13
- shift by 5
- shift by 20
- shift by 13
- ...

For instance suppose the plaintext is

```
It's a dangerous business, Frodo, going out your door.
```

I change everything to lowercase and throwing away non a-z and our plaintext `x` is:

```
itsadangerousbusinessfrodogoingoutyourdoor
```

Suppose the encrypted text is

$$y = \mathtt{y}_1\mathtt{y}_2\mathtt{y}_3\mathtt{y}_4\mathtt{y}_5\mathtt{y}_6\cdots$$

Then

- $\texttt{y}_1$ = shift `i` by 5 = `n`
- $\texttt{y}_2$ = shift `t` by 20 = `n`
- $\texttt{y}_3$ = shift `s` by 13 = `f`
- $\texttt{y}_4$ = shift `a` by 5 = `f`
- $\texttt{y}_5$ = shift `d` by 20 = `x`
- $\texttt{y}_6$ = shift `a` by 13 = `n`
- $\texttt{y}_4$ = shift `n` by 5 = `s`
- $\texttt{y}_5$ = shift `g` by 20 = `a`
- $\texttt{y}_6$ = shift `e` by 13 = `r`
- ...

i.e.,

$$y = \texttt{nnffxnsar}...$$

Exercise 202.10.1. What is the size of the set of all possible keys for the Vigenere cipher? Is this better than the shift cipher?

File: vigenere-attack.tex

202.11 Attacking the Vigenere Cipher

Suppose the length of the key is m. Then you can think of your message as being chopped up into m strings and each is encrypted by the shift of a character and then everything is put together. For instance if the ciphertext is

$$y = \mathtt{y}_1\mathtt{y}_2\mathtt{y}_3\cdots$$

and the key is `fun`, i.e. m is 3, then the 3 strings are

$$\begin{aligned} z_1 &= \mathtt{y}_1\mathtt{y}_4\mathtt{y}_7\cdots \\ z_2 &= \mathtt{y}_2\mathtt{y}_5\mathtt{y}_8\cdots \\ z_3 &= \mathtt{y}_3\mathtt{y}_6\mathtt{y}_9\cdots \end{aligned}$$

Now, the above of course is the encryption process and you know that the key has length 3, therefore you are three different shifts. If you do know the key has length 3, then each of the above z_i is encrypted using the same shift and therefore you can use frequency analysis to decrypt each of them separately.

The problem is when you do *not* have the length of the key. So the first step is always to figure out the length of the key, i.e. m.

METHOD 1. Here's a strategy. Look at the substrings of the ciphertext of length 3. Think about it. Suppose you see the following in your ciphertext:

$$\ldots\mathtt{etu}\ldots\mathtt{etu}\ldots$$

What can you do? One might guess that if the distance between the first `e` to the next is d, then m divides d right? Think about it. So you need to scan for all possible substrings of length 3 and look for such distances. Suppose you have a bunch of such distances $d_1, d_2, \ldots, d_k$. Then m must divide all these d_i's. This means that m divides $\gcd(d_1, d_2, \ldots, d_k)$. Obviously the more d_i's you have the better.

Exercise 202.11.1. Write a function (in your favorite programming language). If returns a hashtable of frequencies of strings of length 1.

Exercise 202.11.2. Generalize the above: Write a function (in your favorite programming language) that accepts a string s and returns a hashtable of where the key-value pair (k, v) has a string length length k for key and the value v is the count of occurrences of k in s.

Exercise 202.11.3. Now modify the above: Write a function (in your favorite programming language) that accepts a string s and returns a hashtable of where the key-value pair (k, v) has a string length length k for key and the value v is a list of index positions in s where the k was found.

Exercise 202.11.4. Using the above, write a function that computes a probabilistic guess on the length of the key when given a string encrypted using the Vigenere cipher.

Once you have a probabilistic guess of the length, say m, you break up the encrypted string into m pieces. Each piece is a a shift ciphertext. So you just use whatever we talked about in the sections on shift cipher.

Done!

Method 2. Here's another technique. It involves some simple counting from discrete math. Suppose again the length of the key of m and you cut up your string into m pieces so that the characters in each substring is shifted by the same amount. Clearly the probabilities of the letters for each substring is preserved. In other words if `e` is encrypted as `g` for the first string and to `n` for the second, etc., then the probability of `g` in the first string and of `n` in the second string must be approximately the same as the probability of `e`. On the other hand, if we assumed that the length wrongly, say $m + 1$, then the probabilities must be different. As a matter of fact the substrings would appear very random. OK. So let's do some math.

Suppose the frequencies of the letters in a string s are f_0, f_1, …, f_{25}, i.e., f_0 is the frequency of `a`, etc. Suppose the length of the s is n. Therefore

$$\sum_{i=0}^{25} f_i = n$$

In this string s, the probability p_0 that a randomly chosen character of s is the character `a` is just f_0/n. Etc. In the following I might write $f(0)$ instead of f_0. So in the following discussion if you see p_0 you can think of f_0/n.

In python or C++, for statistical computation of frequencies of characters (or substrings), you can use a hashtable. (Details in assignment.) For instance in python you can do this. A hashtable in python is called a python dictionary.

In C++ a hashtable is called an unordered map. Try this for python:

```
s = "tobeornottobethatisthequestionandtheanswerisfortytwo"
f = {}
for c in s:
    if c not in f:
        f[c] = 0
    else:
        f[c] += 1
for c,count in f:
    print(c, count)
```

or

```
s = "tobeornottobethatisthequestionandtheanswerisfortytwo"
f = {}
for c in s:
    f[c] = f.get(c, 0)
for c,count in f:
    print(c, count)
```

Make sure you run the above. (For those of you who have taken CISS350, check my notes on hashtables and review `std::unordered_map`.) A lot more information on python dictionary is found in the assignment. Moving on ...

Now the probability of two randomly chosen characters being the same must be

$$I(s) = \frac{\binom{f_0}{2} + \ldots + \binom{f_{25}}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}$$

$\binom{n}{k}$ is the n–choose–k binomial coefficient, i.e. $n!/((n-k)!k!)$. In particular $\binom{n}{2} = n(n-1)/2$.

One thing nice about the above formula is that if any substitution has been applied to the string, *the above value remains the same.*

Exercise 202.11.5. Approximate $I(s) = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)}$ where $p_i = f_i/n$'s are the probabilities taken from our table of letter probabilities.

0.065

So $I(s)$ is approximately $\sum_{i=0}^{25} p_i^2$ where p_i is the probability of a character of the string being the i–th letter.

On the other hand if there is no pattern and everything is random, for instance you cut up your ciphertext, which was encrypted by a substitution with a string of length m, into $m+1$ substrings, then the probabilities of the characters would be almost the same since the string is gibberish. In other words, the probabilities would be about $f_i/n = 1/26$.

Exercise 202.11.6. Suppose s is a random string, i.e. $f_i/n = 1/26$. Compute $I(s) = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)}$. 0.038

Exercise 202.11.7. Write a function `I` that accepts a string of lowercase letters or a list of numbers 0..25 and computes the I–value as described above.

So the question is this: Is the I value sufficiently different for a meaningful string and a random string? Is so, then we have an algorithm for determining m:

Test $m = 1$: You get one piece (the complete string) from the ciphertext y. Compute $I(y)$. If $I(y)$ is approximately 0.065, $m = 1$.

Test $m = 2$: Cut up your ciphertext y into two pieces z_1, z_2. Compute $I(z_1)$, $I(z_1)$. If they are approximately 0.065 (you can use the average of $I(z_1)$ and $I(z_1)$), then $m = 2$.

Test $m = 3$: Cut up your ciphertext y into three pieces z_1, z_2, z_3. Compute $I(z_1)$, $I(z_2)$, $I(z_3)$. If they are approximately 0.065 (you can use the average of all three), then $m = 3$.

Etc.

So now we know how to compute the length of the key. Suppose the plaintext is x and the m pieces of x are $x_1, x_2, x_3, ..., x_m$. Recall y is the ciphertext and the m pieces of y are $z_1, z_2, z_3, ..., z_m$. Suppose the shifts are by $k_1, k_2, \ldots k_m$. So z_1 is a shift of x_1 by k_1, z_2 is a shift of x_2 by k_2, etc.

Suppose p_0 is the probability of `a`, p_1 is the probability of `b`, etc. in unencrypted English. Look at y_1. If you look at the probability of `a` in z_1, then you're really looking at the p_α where α is encrypted as `a`. For instance suppose the shift is by 3. In other words x is encrypted as `a`. So the probability of `a` in z_1 is the same as the probability of `x` which is p_{26-3}. For simplicity, from now on all the indices will be considered mod 26. So when I write p_{-3} I really mean p_{26-3}. In general, if the shift is k, then the probability must be p_{-k}. Similarly, the probability of `b` in y_1 must be p_{1-k}. In general, the probability of the i–th character in z_1 must be p_{i-k}.

Now we make the following observation: What is the probability that a randomly chosen character from z_1 and a randomly chosen character from z_2 are both `a`? It is approximately

$$M(y_i, y_j) = \sum_{\ell=0}^{25} p_{\ell-k_1} p_{\ell-k_2}$$

For any z_i and z_j, this is then

$$\sum_{\ell=0}^{25} p_{\ell-k_i} p_{\ell-k_j}$$

You now notice that the above is the same as

$$\sum_{\ell=0}^{25} p_{\ell} p_{\ell-k_j+k_i}$$

or

$$\sum_{\ell=0}^{25} p_{\ell} p_{\ell+k_i-k_j}$$

The number $k_i - k_j$ is called the **relative shift** of z_i and z_j. relative shift

Exercise 202.11.8. Using the probabilities from our table of letter probabilities, complete the following table:

Relative shift	M
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

You notice that when there is no shift, the M value is approximately 0.065. Otherwise it is much smaller, around 0.04. So this what you can do. Take z_1 and z_2 as examples. Compute the M value for z_1 and z_2. Next you shift z_2 by 1 to get $E_1(z_2)$ (E_1 is the encryption for the shift cipher). Now compute the M value for y_1 and $E_1(y_2)$, i.e. $M(y_1, E_1(y_2))$. Next you compute $M(y_1, E_2(y_2))$. Continue until you have the M value of y_1 and $E_{25}(y_1)$. If the M value for y_1 and $E_{17}(y_2)$ is approximately 0.065, then the relative shift is very likely 17. In other words

$$k_1 - k_2 = 17$$

Now perform the same for different pairs of y_i, y_j. You have $\binom{m}{2}$ pairs and you hope to get as many equations as possible.

You should be able to write your shifts in terms of one shift, say the first k_1. For instance the second shift might be $k_1 + 3$, the third is $k_1 + 2$, etc. You still have the unknown k_1.

But that's only one unknown!!!

You can then easily solve it with 26 different values for k_1. Alternatively, for a Vigenere cipher, the shifts come from a sequence of letters and the letters usually make up a meaningful word. You can scan for meaningful words using your sequence of shifts. For instance if the shifts are $(k_1, k_1 + 17, k_1 + 4, k_1 + 21, k_1 + 10)$, then when $k_1 = 9$ you get the word `janet`.

Notice that the cryptanalysis as above already hints at the fact that algebra and probability theory are extremely important in this area of study. In particular, there is an area of study called Information Theory that studies the amount of randomness or lack of information in random variables encoded within something called the entropy of random variables. Using information theory one can prove that a substitution ciphertext has a unique decryption if the string has a length of at least 25.

In case the above proof hurts your head too much, let me give you an informal proof.

Suppose I have a sequence of numbers:

$$[0.1, 0.2, 0.3, 0.4]$$

Think of these as the probabilities of a language that involves only 4 letters, say the probabilities of the symbols a, b, c, d. Now I rotate the numbers in a circle by 1 step to get this:

$$[0.4, 0.1, 0.2, 0.3]$$

and by 3 steps to get this:

$$[0.2, 0.3, 0.4, 0.1]$$

These corresponds to a shift cipher of 1 and of 3. The relative shift between the 1st and 2nd is 2 steps: you can see that 0.1 is 2 steps away between the two sequences. One way to compute the relative shift (i.e. 2) is to compute the sum of products of the corresponding terms:

```
Relative shift the second by 0
[0.4, 0.1, 0.2, 0.3]
[0.2, 0.3, 0.4, 0.1]
--------------------
0.4*0.2 + 0.1*0.3 + 0.2*0.4 + 0.3*0.1 = 0.2200...

Relative shift the second by -1
[0.4, 0.1, 0.2, 0.3]
[0.3, 0.4, 0.1, 0.2]
--------------------
0.4*0.3 + 0.1*0.4 + 0.2*0.1 + 0.3*0.2 = 0.2399...

Relative shift the second by -2
[0.4, 0.1, 0.2, 0.3]
[0.4, 0.1, 0.2, 0.3]
--------------------
0.4*0.4 + 0.1*0.1 + 0.2*0.2 + 0.3*0.2 = 0.3000...

Relative shift the second by -3
[0.4, 0.1, 0.2, 0.3]
[0.1, 0.2, 0.3, 0.4]
--------------------
0.4*0.1 + 0.1*0.2 + 0.2*0.3 + 0.3*0.4 = 0.2399...
```

You see that the largest value gives you the relative shift.

Of course if the probabilities were from ciphertexts, the numbers will only be *close* to $0.1, 0.2, 0.3, 0.4$ if the lengths are large enough.

[By the way, the reason why you get the largest value when the numbers line up nicely is because the above sum of product of corresponding term is called the inner product and the inner product is largest when the two sequence of number, think vectors, are pointing in the same direction.]

Exercise 202.11.9. Here's another example. Say this is the plaintext:

```
This is a test. This tutorial is aimed at getting
familiar with the bare bones of LaTeX. First, ensure
that you have LaTeX installed on your computer (see
Installation for instructions of what you will
need). We will begin with creating the actual source
LaTeX, and then take you through how to feed this
through the LaTeX system to produce quality output,
such as postscript or PDF.
```

We use the key of `fun` to obtain the ciphertext

```
ybvxcffnrxngmcfyogtlvffvxuvryqfntjngnhtkuznfvfljnnuy
brguejvbsyftzyfnrczvwmgjhfzlrybnysbzbnayyfnrccaxnnqf
riiadihwwbrjhyyexyrnhfyuyqugniakienhfylhhnvthftzjmug
dihbcyqhrjxjjqvqfojavsqvybpwynycalnujupyonqmbzlpjfny
ykfhqybrsnnpyltogmlbzaumijyisjyqybvxnuwihlbgmyyfnrcm
lxnrrnbulbiopjkhffvysbznczwnfzwufmctmgxwenjgtlciz
```

The key `fun` gives us the shifts $(5, 20, 13)$. Let's try to re-discover the shifts.

The (sorted) incidence values I for up to length 6 are

```
Average I-values     keylength
-------------------- ---------
0.06467025914470374  3
0.06262443438914027  6
0.04968484284445197  1
0.04938482570061517  4
0.04908514013749339  5
0.048510313216195575 2
```

Therefore we conjecture that the length is 3. (It shouldn't be surprising that multiples of 3 will also give high I-values, right?) In more details, for the case of testing a keylength of 3, the 3 strings are:

```
yxfxmytfxrfjnknfnygjstfcwjzyyzafcxqidwryxnyqnknyhttm
dbqjjqjsywyljyqzjyfysptmzmyjyxwlmfcxruijfyzzzftxnti

bcnncolfuynnhuflnbuvyznzmhlbsbyncnfiiwjyyhuuiihlnhzu
ichxqfaqbycnuomlfyhbnyolaiiybnibynmnnlokfsnnwmmwjlz

vfrgfgvvvqtgtzvjurebfyrvgfrnbnyranrahbherfygaefhvfjg
hyrjvovvpnaupnbpnkqrnlgbujsqvuhgyrlrbbphvbcfucgegc
```

with respective I-values

```
0.0651056539121
0.0687226346849
0.0601824888371
```

The I–values are computed with

$$\frac{\sum_{i=0}^{25} f(i)f(i-1)}{n(n-1)}$$

where f is the frequency function for the string (for instance $f(0)$ is the frequency of `a`) and n is the length of the string. The average of the above 3 I-values for keylength 3 is 0.0646702591447.

Next we compute the M-values. This means that for each distinct pair of strings (from the 3), let f_1 and f_2 be their frequencies and n, n' be their length, we compute

$$\frac{\sum_{i=0}^{25} f_1(i)f_2((i-g) \bmod 26)}{nn'}$$

with all possible relative shifts $g = 0, 1, 2, ..., 25$. ($f(0)$ means the frequency of `a` in the first string, etc.)

```
M-values        Index of 1st string Index of 2nd string Relative shift g
--------------- ------------------- ------------------- ----------------
0.0498633235932                   1                   2                0
0.0344990102743                   1                   2                1
0.0310114054105                   1                   2                2
0.0330851164106                   1                   2                3
0.0452446036384                   1                   2                4
0.0342162315016                   1                   2                5
0.0333678951833                   1                   2                6
0.0332736355924                   1                   2                7
0.0331793760015                   1                   2                8
0.0291262135922                   1                   2                9
0.0398718069564                   1                   2               10
```

0.0707889527759	1	2	11
0.044584786502	1	2	12
0.0329908568197	1	2	13
0.0328023376379	1	2	14
0.0415684795928	1	2	15
0.0334621547742	1	2	16
0.0401545857291	1	2	17
0.0454331228202	1	2	18
0.031576962956	1	2	19
0.0286549156377	1	2	20
0.0325195588651	1	2	21
0.0441134885475	1	2	22
0.0446790460929	1	2	23
0.0419455179565	1	2	24
0.0379866151381	1	2	25
0.0355035217971	1	3	0
0.0272225395012	1	3	1
0.0371216447744	1	3	2
0.0509232819341	1	3	3
0.046925566343	1	3	4
0.0324576432515	1	3	5
0.0341709499334	1	3	6
0.0503521797068	1	3	7
0.0397867885018	1	3	8
0.0293165810013	1	3	9
0.0390253188654	1	3	10
0.0426422996383	1	3	11
0.0329335617742	1	3	12
0.0278888254331	1	3	13
0.0438796877974	1	3	14
0.0348372358652	1	3	15
0.028745478774	1	3	16
0.0399771559109	1	3	17
0.0667237768894	1	3	18
0.0402627070246	1	3	19
0.029602132115	1	3	20
0.0323624595469	1	3	21
0.0442604226156	1	3	22
0.0345516847516	1	3	23
0.0413097277746	1	3	24
0.037216828479	1	3	25
0.0413097277746	2	3	0
0.035313154388	2	3	1
0.0286502950695	2	3	2
0.0434989529792	2	3	3
0.0369312773653	2	3	4
0.0317913573196	2	3	5
0.0440700552065	2	3	6
0.0655815724348	2	3	7
0.0368360936608	2	3	8
0.031505806206	2	3	9
0.0355987055016	2	3	10
0.0369312773653	2	3	11
0.037216828479	2	3	12
0.0395012373882	2	3	13
0.0356938892062	2	3	14
0.0318865410242	2	3	15
0.0336950314106	2	3	16
0.0343613173425	2	3	17
0.055396916048	2	3	18
0.0403578907291	2	3	19
0.04016752332	2	3	20
0.0357890729107	2	3	21
0.0484485056158	2	3	22

```
0.0339805825243                  2                   3                  23
0.0326480106606                  2                   3                  24
0.0328383780697                  2                   3                  25
```

We sort and list 6 rows with the highest M-values:

```
M-values        Index of 1st string Index of 2nd string Relative shift g
--------------- ------------------- ------------------- ----------------
0.0707889527759                  1                   2                11
0.0667237768894                  1                   3                18
0.0655815724348                  2                   3                 7
 0.055396916048                  2                   3                18
0.0509232819341                  1                   3                 3
0.0503521797068                  1                   3                 7
```

For the first row, if the M-value using index 1 and index 2 string (i.e., z_1, z_2) with a relative shift of 11. The computed M–value is 0.0707.... In other words if f_1 and f_2 denote the frequency data of z_1 and z_2 respectively, then the M–value shown in the first row above is

$$\frac{\sum_{i=0}^{25} f(i)f'((i-11) \bmod 26)}{n_1 n_2}$$

where n_1 and n_2 are the lengths of z_1, z_2 respectively and $f_1(0)$ is the frequency of a in z_1, $f_1(1)$ is the frequency of b in z_1, etc. From the top 2 entries of M-values:

```
M-values        Index of 1st string Index of 2nd string Relative shift g
--------------- ------------------- ------------------- ----------------
0.0707889527759                  1                   2                11
0.0667237768894                  1                   3                18
```

we get

$$\begin{aligned} k_1 - k_2 &\equiv 11 \pmod{26} \\ k_1 - k_3 &\equiv 18 \pmod{26} \end{aligned}$$

or

$$\begin{aligned} k_2 &\equiv k_1 - 11 \equiv k_1 + 15 \pmod{26} \\ k_3 &\equiv k_1 - 18 \equiv k_1 + 8 \pmod{26} \end{aligned}$$

At this point, even though we don't have the 3 shifts yet, we at least know that the shifts is probably of the form

$$(k_1, \quad (k_1 + 15) \bmod 26, \quad (k_1 + 8) \bmod 26)$$

There is now *one* unknown. Now trying different values of k_1, we see that when $k_1 = 5$, we get

$$k_1 = 5, \quad k_2 = 20, \quad k_3 = 13$$

which corresponds to the word `fun`. Vóila! ... we have discovered the key.

Exercise 202.11.10. Here are things you need to do right away.

1. Given a string y (a ciphertext) and an integer m (say 3) How to you form z_1, z_2, z_3 in python?
2. Given a string z write a function (call if `freq`) so that `freq(z)` returns the frequency data of z as a dictionary.
3. Write a function `avg_I` such that `avg_I(y, m)` computes the average I–value of the m pieces $z_1, z_2, ..., z_m$ of y.
4. Write a function `estimate_m` such that `estimate_m(y)` gives as estimate for m the length of key by doing a for-loop on $m = 1, 2, 3, ...$ until a large enough I value is attained and at that point the m is returned. You might want to write it with two other paramters, i.e., `estimate_m(y, m_cutoff=20, I_cutoff=0.06)` so that when the average I is ≥ 0.06, the m is returned. The function returns `None` if m reaches `m_cutoff`.
5. Write a function `M` such that `M(freq1, freq2, g)` computes the M–value for the obvious data.
6. Write a function `relative_shifts` such that `relative_shifts(freqs)` return a tuple of $m - 1$ value for the relative shifts. (For instance in the example above, when $m = 3$, the relative shifts are 15 and 8.
7. Finally once you call `relative_shifts` you have the $m - 1$ relative shifts. So a for-loop for k_1 over a..z and print all the 26 words using the relative shifts and for the shifts, print the keys (as a word) and print the decryption of the first (say) 40 characters. The output with the meaning key and the meaningful decryption will give you the key. (This last part can be automated if you have a dictionary of English words and you know how to perform word segment – see CISS358.)

File: substitution.tex

202.12 Substitution Cipher

OK folks, things are getting more fun (or worse, depending on your point of view). If you've read your notes and following the readings you should know why the substitution cipher is not that easy to break. Right?

Because of the gap in the probability between `e` and the next letter, you can usually figure out what `e` encrypts to. The problem is that next group of letters have frequencies which are rather close together.

Here's a suggestion. If you know (or think you do) that `e` is encrypted to `r`, then you should use the digrams. Basically the idea is the same as solving crossword puzzles. You want to decrypt the letters near to those letters which are already decrypted, just like in crossword puzzles you want to work near to whose rows and columns that are already solved. So look at all the places where `r` are decrypted as `e`. Look at the letter before and after it and hope that you can now decrypt by looking at digrams in the ciphertext of the form `?r` and `r?`

File: block-and-stream-ciphers.tex

202.13 Block and Stream Ciphers

Recall that in Caesar cipher, you only define the encryption and decryption for single characters.

The encryption/decryption is then extended to a whole string by encrypting/decrypting character-by-character.

So if you're encrypting the string `cat` with the encryption function E_K, you just do

$$E_K(\texttt{cat}) = E_K(\texttt{c})E_K(\texttt{a})E_K(\texttt{t})$$

Another thing you should know is that, instead of writing

$$\texttt{cat}$$

might write

$$\texttt{c}||\texttt{a}||\texttt{t}$$

i.e., it's common to use || to denote concatenation of data.

A **block** cipher is a cipher system where more than one character (a block) is encrypted together at the time same.

But why encrypt a block of characters at a time?

In the case of shift and affine cipher, the frequency of the plaintext character is the same as the frequency of the corresponding ciphertext character. Sure, an `a` might become a `t` which looks different from `a`. But guess what? `a`'s fingerprint – its frequency – follows it to the ciphertext.

The same idea is the same for Vigenere as long as you have the length of the key.

The next cipher, Hill cipher, is difference. We're going to encrypt several characters at the same time. FOr instance Hill cipher (depending on the key) can encrypt 2 characters at the same time to produce 2 new characters.

Exercise 202.13.1. Is Caesar's cipher a block or stream cipher?

File: hill-cipher.tex

202.14 Hill's cipher

Hill's cipher

Hill's cipher was announced in 1929. This cipher works with a block of n characters at a time. For both encryption and decryption, you treat each block of n characters (or mod 26 integers) as a column vector and we multiply it with an n–by–n matrix. Of course everything is done in mod 26. Here are more details.

Recall how to multiple a n–by–n matrix with a column vector of size n. I'll do this for $n = 2$.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

(If you have not seen this before take the computer graphics class CISS380. You can't call yourself a serious computer scienctist if you can't do matrix multiplication. Most top CS programs actually has a linear algebra requirement.) That's matrix multiplication where all numbers are in $\mathbb{R}$. The (multiplicative) **inverse matrix** of

inverse matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

The expression $ad - bc$ is called the **determinant** of $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and is written

determinant

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

If a matrix has an inverse, we say that the matrix is **invertible** .

invertible

Not only can you multiply a 2-by-2 matrix with a 2-by-1, you can actually also multiply a 2-by-2 matrix with a 2-by-2. Here's how you do it:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} aA + bB & aC + bD \\ cA + dB & cC + dD \end{pmatrix}$$

Exercise 202.14.1. The following refers to matrices with $\mathbb{R}$ values. Do all these exercises:

1. Compute
$$\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$
2. Compute
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}$$
3. If M, N are 2-by-2 matrices, is it always true that $MN = NM$?
4. Prove the following theorem: Let A, B, C be three 2-by-2 matrices. Prove that
$$(AB)C = A(BC)$$

$\square$

Define the following:
$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
This is called the (2-by-2) **identity matrix**

identity matrix

Exercise 202.14.2. The following refers to matrices with $\mathbb{R}$ values.

1. Show that
$$I \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} I$$
2. Show that
$$I \begin{pmatrix} 2 \\ 4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$
3. Compute
$$\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}^{-1}$$
Then show that
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$

$\square$

Exercise 202.14.3. The following refers to matrices with $\mathbb{R}$ values. Let M be a 2-by-2 matrix.

1. Is every matrix invertible?

2. Prove that if M is a 2-by-2 matrix,
$$MM^{-1} = I = M^{-1}M$$
3. Prove that if M is a 2-by-2 matrix, Prove that
$$IM = M = MI$$
4. Prove that if v is a 2-by-1 matrix,
$$Iv = v$$

$\square$

The set of n–by–n matrices with $\mathbb{R}$ values is usually denoted by $\mathrm{M}_n(\mathbb{R})$. The set of invertible matrices is usually denoted by $\mathrm{GL}_n(\mathbb{R})$. This is called the **general linear group** of n-by-n matrices over $\mathbb{R}$.

It turns out that everything works just as fine if the numbers are in mod 26. You just need to be careful with the determinant of the inverse computation. Recall that
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$
for values in $\mathbb{R}$. This is the same for values in $\mathbb{Z}/26$ as long as you think of
$$\frac{1}{ad-bc}$$
as the multiplicative inverse of $ad - bd$ in $\mathbb{Z}/26$.

Exercise 202.14.4. The following refers to matrices with $\mathbb{Z}/26$ values. All the values in the matrices should be simplied to a value in the range 0..25. Do all these exercises:

1. Compute
$$\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$
2. Compute
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}$$
3. If M, N are 2-by-2 matrices, is it always true that $MN = NM$?
4. Prove the following theorem: Let A, B, C be three 2-by-2 matrices. Prove

that

$$(AB)C = A(BC)$$

(Look at the facts below on breaking the Hill's cipher. Why did I ask you to prove this theorem? Where is this fact used below?)

5. Does

$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$

have an inverse? If it does compute it. If not explain why.

6. Does

$$\begin{pmatrix} 1 & 2 \\ 23 & 7 \end{pmatrix}$$

have an inverse? If it does compute it. If not explain why.

7. Compute

$$\begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix}^{-1}$$

and show that

$$\begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix}^{-1} \begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix} = I = \begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix}^{-1}$$

□

Exercise 202.14.5. Say the key is

$$M = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$$

Here $n = 2$, i.e., the block size is 2.

1. What is the encryption of $x =$ `fortytwo`? Call the ciphertext y.
2. What is the inverse matrix M^{-1} of M? (Note: we are in mod 26.) Check that $MM^{-1} = I = M^{-1}M$.
3. Decrypt your y and make sure you get x.
4. Look for the probability of 2-grams. Take a long enough string, encrypt it using Hill cipher and with the above key. Look at the probability of 2-grams in the ciphertext. Compare the two probabilities of the 2-grams from the plaintext and from teh cipherptext.

This should not be surprising: The set of n–by–n matrices with mod 26 values is usually denoted by $\mathrm{M}_n(\mathbb{Z}/26)$. The set of invertible matrices is usually denoted by $\mathrm{GL}_n(\mathbb{Z}/26)$.

Exercise 202.14.6. Let

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

a mod 26 matrix. Prove that

$$MI = M = IM$$

for any 2-by-2 mod 26 matrix M. □

Note that in the above, since the block size is 2, you would need to encrypt a string with a string length that is a multiple of 2. If it's not, you can (for instance) pad with some character.

The important thing to observe is that if

$$E(M, (x, y)) = (x', y')$$

(M is a matrix key) note that x takes part in the value x' *and* y'. Same thing for y'. This means that if I change one character in my plaintext, *two* character in the ciphertext is changed. Informally this means that the statistical behavior of x is spread out across multiple characters. Note only that x' also depends on several values of the key (the matrix). This makes it much hard to analyze the key one part at a time when studying ciphertexts. This makes it a lot harder to break Hill's cipher, especially when the matrice size is larger, say the matrix is 32-by-32.

The above two important observations were later formalized and studied by Claude Shannon in 1945.

1. The fact that a character of the ciphertext depends on several parts of the key is the concept of **confusion** and confusion
2. the fact that one character in the plaintext influences multiple characters in the ciphertext is called the concept of **diffusion**. diffusion

More on that in a much later chapter.

Exercise 202.14.7. Write a function `matmult` so that if `M` is a 2D array of size 2-by-2 and `v` is a column vector, i.e., a 2D array of size 2-by-1, then `matmult(M, v)` will return the product `Mv` as described above. Remember to mod by 26 all the values. (If you know your linear algebra – either from computer graphics CISS380 or linear algebra math class, you can figure out how to complete this function for `M` and `v` of sizes n-by-n and n-by-1 respectively)

Exercise 202.14.8. Use a 2D array for 2-by-2 matrices. Write a function `matdet` so that `matdet(M)` return the determinant of `M`. Remember that we are in mod 26. □

Exercise 202.14.9. Use a 2D array for 2-by-2 matrices. Write a function `matinv` so that `matinv(M)` return the inverse matrix of `M`. Return `None` if `M` is not invertible. Remember that we are in mod 26. □

Exercise 202.14.10. Let

$$M = \begin{pmatrix} 3 & 7 \\ 5 & 2 \end{pmatrix}$$

1. Encrypt $x =$ `solongandthanksforthefish` using Hill's cipher with key M. Call the resulting string `y`.
2. Compute M^{-1}.
3. Decrypt y. You should get x.

Exercise 202.14.11.

1. What is the size of the key space assuming the matrix is 2-by-2? (I mean for mod 26 of course.) This means: how many invertible mod 26 matrices are there? Or: What is $|\operatorname{GL}_2(\mathbb{Z}/26)|$? You can write down a few invertible mod 26 matrices. But after a while you might want to write a program.
2. See if you can find a plausible formula for the case of $n = 2$. The formula involves 2 and 13 because $26 = 2 \times 13$ and because $n = 2$. (Yes, there's a formula.) It also works if mod 26 is replaced by mod N for any positive integer $N \geq 1$. You can try to work with matrices with values in $\mathbb{Z}/5$ then $\mathbb{Z}/8$, etc. and see if you can find a formula for $|\operatorname{GL}_2(\mathbb{Z}/N)|$. (The formula works when n is any integer ≥ 1. But you would need to know about matrices of size larger than 2-by-2.)

□

Collecting up all the information above, formally, the Hill's cipher of block size n (just think of $n = 2$) is defined as follows: Let

$$P = C = \{0, ..., 25\}^n = (\mathbb{Z}/26)^n$$

and

$$K = \operatorname{GL}_n(\mathbb{Z}/26)$$

The encryption function E

$$E : K \times P \to C$$

is defined to be

$$E(M, x) = Mx$$

and

$$D : K \times C \to P$$

is defined to be

$$D(M, x) = M^{-1}x$$

And (E, D) is a cipher because

$$D(M, E(M, x)) = D(M, Mx) = M^{-1}Mx = Ix = x$$

Exercise 202.14.12. What about 3-by-3 matrices? How do you compute the inverse? What is the size of the key space for 3-by-3? What about n–by–n?
$\square$

BREAKING HILL'S CIPHER

Now how do you break the Hill's cipher?

Let's stick to a 2-by-2 key. Suppose you have the following: you have the two plaintexts $x_1 = (A, B)$ and $x_2 = (C, D)$ and their encryptions $y_1 = (A', B')$ and $y_2 = (C', D')$. Therefore

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} aA + bB \\ cA + dB \end{pmatrix} = \begin{pmatrix} A' \\ B' \end{pmatrix}$$

and

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} C \\ D \end{pmatrix} = \begin{pmatrix} aC + bD \\ cC + dD \end{pmatrix} = \begin{pmatrix} A' \\ B' \end{pmatrix}$$

In the above, the unknowns are a, b, c, d, i.e., the key. These two equations can be combined to get an equation involving the product of 2-by-2 matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} aA + bB & aC + bD \\ cA + dB & cC + dD \end{pmatrix} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix}$$

You now have this matrix equation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix}$$

Note that you have the values of $A, B, C, D, A', B', C', D'$. The unknowns are a, b, c, d. What do you do? You multiply on the right by the inverse matrix of $\begin{pmatrix} A & C \\ B & D \end{pmatrix}$ to get

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1}$$

which gives you

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} I = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1}$$

which finally gives you the key

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1}$$

There's an assumption above: we have to assume that $\begin{pmatrix} A & C \\ B & D \end{pmatrix}$ is invertible. It's possible the first 4 characters of the original message does not form an invertible matrix. But if you have enough pairs of plaintext and corresponding ciphertext blocks, you hope to be able to get an invertible matrix and perform the above computation. Therefore we are assuming the known plaintext attack model. Or we are assuming the chosen plaintext attack model.

`File: permutation-cipher.tex`

202.15 Permutation cipher

In the above ciphers, a character is replaced by another. The **permutation cipher** is different: each character is *moved* to a different *location* in the plaintext to form the ciphertext. permutation cipher

Here's an example. Look at this:

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 1 & 2 & 5 & 4 & 6 \end{pmatrix}$$

It's just the function where a value in the top row maps to the corresponding value at the bottom row:

$$\begin{aligned} \pi(1) &= 3 \\ \pi(2) &= 1 \\ \pi(3) &= 2 \\ \vdots &\quad \vdots \\ \pi(6) &= 6 \end{aligned}$$

Let π is called a permutation of $\{1,2,3,4,5,6\}$. A **permutation** is simply a bijection (1–1 and onto) function from $\{1,2,3,...,n\}$ to $\{1,2,3,...,n\}$. As a shorthand, the above permutation permutation

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 1 & 2 & 5 & 4 & 6 \end{pmatrix}$$

is also written as

$$\pi = (1\ \ 3\ \ 2)(4\ \ 5)(6)$$

This is called the **cycle notation** of π. In the above example, we say that the permutation π has **length** 6. cycle notation length

Exercise 202.15.1. What is the cycle notation of

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 1 & 7 & 2 & 5 & 8 & 3 & 6 \end{pmatrix}$$

How does a permutation give you an encryption? For the permutation $\pi = (1\ \ 3\ \ 2)(4\ \ 5)(6)$, you do this:

1. character at position 1 goes to position 3,
2. character at position 2 goes to position 1,
3. etc.

In the example above, the n is 6. For instance for the string `marvin` and the permutation $\pi = (1\ \ 3\ \ 2)(4\ \ 5)(6)$, the since in the permutation $1 \mapsto 3$, the character at position 1 goes to position 3:

```
1 2 3 4 5 6
m a r v i n
    m
```

since $3 \mapsto 2$, the character at position 3 goes to position 2:

```
1 2 3 4 5 6
m a r v i n
  r m
```

Etc. Altogether I get

$$E(\pi, \texttt{marvin}) = \texttt{armivn}$$

This assumes your string has a length which is a multiple of 6. So you might need to add some dummy data at the end of your string.

Note that in the above example, you have to encrypt substrings of length 5. Therefore if the original plaintext does not have a length that is a multiple of 5, you would need to pad it with some characters until the length is a multiple of 5.

Exercise 202.15.2. How would you choose to pad the plaintext? You might want to finish this section before you answer this question. (I.e., you want to know what Eve will do and design your padding to make it difficult for her.)

The decryption is the same algorithm as the encryption except that the permutation is read in the "opposite direction". If

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$$

then the inverse of π, denoted by π^{-1} is the "opposite" of the above:

$$\pi^{-1} = \begin{pmatrix} 3 & 1 & 2 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

(do you see the row switch?) and to make things look nice, you arrange the top row:

$$\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}$$

Using the cycle notation,

$$\pi^{-1} = (1 \ \ 2 \ \ 3)(4 \ \ 5)$$

So

$$D(\pi, y)$$

is the same as

$$E(\pi^{-1}, y)$$

So

$$\begin{aligned} D((1 \ \ 3 \ \ 2)(4 \ \ 5), \texttt{armivn}) &= E((1 \ \ 3 \ \ 2)(4 \ \ 5)^{-1}, \texttt{armivn}) \\ &= E((1 \ \ 2 \ \ 3)(4 \ \ 5), \texttt{armivn}) \\ &= \texttt{marvin} \end{aligned}$$

Exercise 202.15.3.

1. Encryption `"Where there's life there's hope ...  and need of vittles."` using the permutation cipher with key $\pi = (1\ \ 5)(2\ \ 6\ \ 3\ \ 4)$. Pad with `z` if necessary.

2. What is the inverse of π?

3. Decrypt your ciphertext from the first part.

Exercise 202.15.4. What is the size of the key space of the permutation cipher if you know that the permutation is a permutation on 1, 2, 3, 4, 5? What if it's a permutation on 1, 2, 3, 4, 5, 6, 7, 8, 9, 10?

Exercise 202.15.5. Why is the permutation cipher a special case of the Hill cipher? To try it out, write down an example of Hill cipher and re-interpret it as a permutation cipher. You'll see that the matrix has a special form – it has only 0s and 1s arranged in a certain way. Such a matrix is called a **permutation matrix**.

Breaking the permutation cipher

Frequency analysis obviously won't help: each letter is not replaced by another. It's the position of a letter that is changed.

Now suppose the permutation is

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$$

Suppose you know the length of the permutation is 5. After breaking your ciphertext into substrings of length 5, suppose one of these substrings is

```
hatoe
```

We know that `th` is a commonly occurring digram. Then it's natural to suspect that this comes from the encryption of 5 letters containing `th`, either

```
plaintext : ..... th... ...
ciphertext: ..... hatoe ...
```

or

```
plaintext : ..... .th.. ...
ciphertext: ..... hatoe ...
```

or

```
plaintext : ..... ..th. ...
ciphertext: ..... hatoe ...
```

or

```
plaintext : ..... ...th ...
ciphertext: ..... hatoe ...
```

Whereas for a substring of length 5, there are $5! = 120$ possible permutations, once two is fixed, there are $3! = 6$ possible permutations. This might cut down on the search for the key and help break the ciphertext. If `th` is not found or the above analysis involving `th` does not work, you go on to the next commonly occurring digram.

Of course you do not know if the length of the permutation is length 5. You therefore have to do a loop over all possible permutation lengths. Of course the key length cannot be 1 since that would be doing nothing! For key of size two, the permutation must be $(1, 2)$. For a key of size three, there are $3! = 6$ permutations: There are 6 possible permutations: $(1)(2)(3)$, $(1)(2, 3)$, $(2)(1, 3)$, $(3)(1, 2)$, $(1, 2, 3)$, $(1, 3, 2)$. Since there are not many permutations of length 3, for this case, you might want to simply try all permutations. In general, you try key lengths dividing the length of the ciphertext.

Note that it's also possible that the `th` in the plaintext is broken up with the `t` is one substring of length 5 and `h` in the following substring. For instance, here is a case:

```
plaintext : canth efish
ciphertext: acnht feihs
```

Of course it's also OK if you do not wish to consider cases where the digram is split across two substrings, since in the above case, you would have picked up the `he` when you consider a key of length (say) 10.

By looking at lots of digrams and trigrams, if the ciphertext is long enough, you should be able to break the code.

Example 202.15.1. Let me break the following permutation ciphertext

```
imahsnrwnesotanfusefvitieedoslwswma
```

SOLUTION.

The length is 35. Since the length of the permutation must divide 35, it must be 1, 5, 7, 35.

LENGTH 1. Of course if the length is 1, then the ciphertext is the same as the plaintext. But the plaintext is meaningless. Therefore the length cannot be 1.

LENGTH 5. Assume the key length is 5. I break up the string above into substrings of length 5 to make it easier to read:

```
imahs nrwne sotan fusef vitie edosl wswma
```

LENGTH 5. DIGRAM `TH`. I don't see any `th` in each substrings or consecutive substrings.

LENGTH 5. DIGRAM `HE`. I do see `he`:

```
imahs nrwne sotan fusef vitie edosl wswma
```

If you assume these two characters come from `he`, then it has to come from this:

```
plaintext:  ....h e.... ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which means the permutation contains $1 \mapsto 5, 5 \mapsto 4$. If that's the case, we have

```
plaintext:  s...h e...n n...a f...e e...i l...s a...m
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

Look at just the first group of 5 characters the `s...h` can only be `simah`, `siamh`, `saimh`, `samih`, `smiah`, or `smaih`. The only promising one is `siamh` (maybe "Siam has cats ...?"). Using this we get

```
plaintext:  siamh enwrn nstoa ffsue evtii leods awwsm
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is meaningless.

[force newpage]

[force newpage]

Length 5. Digram in. The next digram to try is `in`:

```
plaintext:  ....i n.... ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

or

```
plaintext:  ....i n.... ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

The first is not possible. (Why?) The second gives us

```
plaintext:  h...i n...n a...s e...f i...v s...e m...w
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

Looking at just the first 5 characters, the only possible decryptions are `hmasi`, `hmsai`, `hamsi`, `hasmi`, `hsami`, `hsmai`. Only `hamsi`, `hasmi`, are promising. From `hamsi`,

```
plaintext:  hamsi nwren atons esuff itiev sodle mwsaw
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is meaningless. From `hasmi`,

```
plaintext:  hasmi nwern atnos esfuf iteiv solde mwasw
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is meaningless.

Length 5. Digram er. The next digram to try is `er`:

- From

  ```
  plaintext:  ..... er... ..... ..... ..... ..... .....
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  we get

  ```
  plaintext:  sm... er... no... fu... ei... ld... as...
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  The first 5 characters can only be `smahs`, `smash`, `smhas`, `smhsa`, `smsah`, `smsha`. Only `smash` seems to form a word. (`smhas` might be for instance a chasm has separated us, but the smhas is the *beginning* of the plaintext.) Using that, we get

```
plaintext:  smash erwen notna fusfe eitei ldols aswam
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is not meaningful.

- From

  ```
  plaintext:  ..... .er.. ..... ..... ..... ..... .....
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  we get

  ```
  plaintext:  .sm.. .er.. .no.. .fu.. .ei.. .ld.. .as..
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  The first 5 characters can only be ismah, ismha, asmih, asmhi, hsmia, hsmai, none of which is meaningful, except possibly for ismah (example: is mahi mahi a fish?), ismha (example: a schism happened overnight), asmih (example: protoplasm i have), asmhi (example: protoplasm have i) but you can check that these do not lead to anything meaningful.

- From

  ```
  plaintext:  ..... ..er. ..... ..... ..... ..... .....
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  we get

  ```
  plaintext:  ..sm. ..er. ..no. ..fu. ..ei. ..ld. ..as.
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  The first 5 characters can only be iasmh, ihsma, aismh, ahsmi, hisma, hasmi. Only ahsmi, hisma and hasmi seem promising. For ahsmi, we get

  ```
  plaintext:  ahsmi wnern tanos sefuf tieiv oslde wmasw
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```

  which is meaningless. For hisma, we get

  ```
  plaintext:  hisma nnerw asnot effus iveit seldo mwasw
  ```

```
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which finally gives us the plaintext

```
hismannerwasnoteffusiveitseldomwasw
```

This gives us

```
His manner was not effusive. It seldom was.
```

The last `w` is redundant (it's a padding).

The key (i.e., permutation) is $(1, 4, 2)(3, 5)$ with a key length of 5. □

Exercise 202.15.6. Of course the process is tedious and error prone. So you know what to do:

- Write a program to encrypt and decrypt messages for the permutation cipher. For instance you can try this permutation

  $$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 4 & 1 \end{pmatrix}$$

  by entering `2,3,5,4,1`. Note that (mathematically speaking) permutations are written as bijection between sets $\{1, 2, 3, ..., m\}$. You might want to start with 0 instead. In that case, you should tell your user.

- From the above experience of computation-by-hand, we see that for breaking a permutation cipher, the program accepts a ciphertext and ask you for a permutation. Furthermore your program should allow you to enter a permutation partially. For instance to enter this permutation

  $$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ & 3 & & & 1 \end{pmatrix}$$

  the user enters `?,3,?,?,1` in your program. You also want to allow the user to modify a permutation.

- You also want the program to list commonly occurring digrams that appears in the ciphertext, especially including those that appear within substrings of length m (the length of the permutation).

- With all the above, you can then automate the process to imitate what I did in the previous example.

Exercise 202.15.7. Decrypt this permutation ciphertext:

```
rhetibbalhotnweerstthigalontakeinuntolfemsorywae
```

and write down the key. It's advisable to do some of these decryption by hand.

Exercise 202.15.8. Decrypt this permutation ciphertext:

```
uyashonlpotlsasdaskeathrsshcdp
```

and write down the key. It's advisable to do some of these decryption by hand.

Exercise 202.15.9. Decrypt this permutation cipher:

```
konusyowairduathsiatirttleistmkewihsihenatprdmpevn
oanigoliorwackatmhfianboemlrteseeudganubtodatofeao
hispaxtinyioenedpipstcheiaatlelairywdhlitsisdntehe
owytmmhaotorlteedmnhieswwanogutyyf
```

and write down the key. It's advisable to do some of these decryption by hand.

Exercise 202.15.10. Decrypt this permutation ciphertext:

```
wshiattesotbtfeistamiweseostwrhttmsoiefd
```

and write down the key. It's advisable to do some of these decryption by hand.

Exercise 202.15.11. You have been intercepting ciphertexts sent between Alice and Bob. You know that they use the permutation cipher. You notice this fact: All the 1000 ciphertexts have length 720. Why is that? Or is that just coincidence?

File: one-time-pad.tex

202.16 One time pad

The one time pad is easy: Suppose Bob wants to send a message. Bob will need to translate this into bits. (For instance use the ASCII code or some other agreed upon format.) Say the plaintext (in bits) is 00101011.

The key is a very long sequence of random bits that Bob and Alice has agreed upon. Suppose the sequence is 01011010111011100010101011100110101001.

Bob takes his message 00101011 and exclusive-or with the first eight bits of the key 0101101011101110001:

```
00101011
01011010111011100010101011100110101001
01110001
```

He then removes the 8 bits used:

~~01011010~~111011100010101011100110101001

and sends the ciphertex (in bits) 01110001 to Alice.

Once Alice received 01110001, she exclusive-or with her key:

```
01110001
01011010111011100010101011100110101001
00101011
```

which is the plaintext. She also removes the bits in her key used in the decryption: ~~01011010~~111011100010101011100110101001.

The next time the communicate, they will start with the remaining bits of their key.

That's it.

It's a vry simple cipher. We don't have enough math (yet) to prove it, but you can sense the this cipher is extremely secure. Why?

Exercise 202.16.1. Suppose the ciphertext is zsdeasheir. (Of course you have to convert this to bits, say using ASCII code. Find a key so the the above is decrypted to killatfour. Find another key so that the above is decrypted to anapplepie.

There's a rumor that during the Cold War, Washington D.C. communicates with Moscow using one time pad.

To be secure the key must be a random sequence. Furthermore, the key cannot be reused. The other problem is that the key is really long.

(Using the concept of entropy of information theory, Claude Shannon can prove that the ciphertext contains no information about the plaintext, other than the length.)

File: lfsr.tex

202.17 Linear Feedback Shift Register

Recall that the one time pad uses exclusive-or to operator on bit sequences. Note that the exclusive-or is the same as addition mod 2!!! In other words addition *in mod 2*

$$\begin{aligned} 0+0 &\equiv 0 \pmod{2} \\ 0+1 &\equiv 1 \pmod{2} \\ 1+0 &\equiv 1 \pmod{2} \\ 1+0 &\equiv 0 \pmod{2} \end{aligned}$$

is the same as exclusive-or operation *on bits*:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 0 &= 0 \end{aligned}$$

(I'm using $\oplus$ for exclusive-or bit operator – that's pretty standard.)

Now I'm going to this: first I define the following bit sequence of length 5:

$$x_1 x_2 x_3 x_4 x_5 = 10110$$

which is the same as defining integer $x_1, ..., x_5$ in $\mathbb{Z}/2$. Then I define

$$x_{n+6} \equiv x_{n+1} + x_{n+2} + x_{n+4} \pmod{2}$$

for $n \geq 0$. For instance

$$x_6 \equiv x_1 + x_2 + x_4 \equiv 1 + 0 + 1 \equiv 0 \pmod{2}$$

Here are next about 15:

$$\begin{aligned}
x_6 &\equiv x_1 + x_2 + x_4 \equiv 1+0+1 \equiv 0 \pmod 2\\
x_7 &\equiv x_2 + x_3 + x_5 \equiv 0+1+0 \equiv 1 \pmod 2\\
x_8 &\equiv x_3 + x_4 + x_6 \equiv 1+1+0 \equiv 0 \pmod 2\\
x_9 &\equiv x_4 + x_5 + x_7 \equiv 1+0+1 \equiv 0 \pmod 2\\
x_{10} &\equiv x_5 + x_6 + x_8 \equiv 0+0+0 \equiv 0 \pmod 2\\
x_{11} &\equiv x_6 + x_7 + x_9 \equiv 0+1+0 \equiv 1 \pmod 2\\
x_{12} &\equiv x_7 + x_8 + x_{10} \equiv 1+0+1 \equiv 1 \pmod 2\\
x_{13} &\equiv x_8 + x_9 + x_{11} \equiv 0+0+1 \equiv 1 \pmod 2\\
x_{14} &\equiv x_9 + x_{10} + x_{12} \equiv 0+0+1 \equiv 1 \pmod 2\\
x_{15} &\equiv x_{10} + x_{11} + x_{13} \equiv 0+1+1 \equiv 0 \pmod 2\\
x_{16} &\equiv x_{11} + x_{12} + x_{14} \equiv 1+1+1 \equiv 1 \pmod 2\\
x_{17} &\equiv x_{12} + x_{13} + x_{15} \equiv 1+1+0 \equiv 0 \pmod 2\\
x_{18} &\equiv x_{13} + x_{14} + x_{16} \equiv 1+1+1 \equiv 1 \pmod 2\\
x_{19} &\equiv x_{14} + x_{15} + x_{17} \equiv 1+0+0 \equiv 1 \pmod 2\\
x_{20} &\equiv x_{15} + x_{16} + x_{18} \equiv 0+1+1 \equiv 0 \pmod 2
\end{aligned}$$

More generally, after defining $x_1, ..., x_5$ (the initial conditions) you can generated the sequence x_i using

$$x_{n+6} \equiv c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} + c_4 x_{n+4} + c_5 x_{n+5} \pmod 2$$

for $n \geq 0$ for constants $c_1, ..., c_5$ in $\mathbb{Z}/2$. I will say that this is linear relation has **degree 5**. Even more generally, you can have any number of bits for the initial condition. Say you begin with $x_1, ..., x_k$ (the initial condition) and the relation is

degree 5

$$x_{n+k+1} \equiv c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} + \cdots + c_k x_{n+k} \pmod 2$$

LSRFs are very easy to implement in both hardware and software and they are extremely fast (they just access bits and XOR them). The LSRF generator itself need to remember the k bits $c_1, ..., c_k$ (which is fixed) and the k bits of the sequence so far $x_{n+1}, ..., x_{n+k}$ in order to generate the next bit x_{n+k+1}.

In the above example, the sequence from x_1 to x_{20} is

10110010001111010110010001111010110...

Notice that the pattern repeats itself:

$$101100100011110 \;\; 101100100011110 \;\; 10110...$$

The period is 15.

The problem with the one-time pad is that you need to generate a random sequence of 0s and 1s. You can see that with 5 bits

$$x_1 x_2 x_3 x_4 x_5 = 10110$$

and the relation

$$x_{n+6} \equiv x_{n+1} + x_{n+2} + x_{n+4} \pmod 2$$

which involves (1,1,0,1,0) (5 bits), a total of 10 bits) we can generate

$$101100100011110$$

which has length 15. The 15 bits is somewhat random – we say that the 15 bits are **pseudorandom**. Therefore LFSR can be used to generate a pseudorandom bit sequence, which can used, for instance, as a key for the one-time pad. Of course you want to find a LSFR with extremely long periods.

pseudorandom

Exercise 202.17.1. What if you define

$$x_1x_2x_3x_4 = 1011$$

and use

$$x_{n+5} \equiv x_{n+1} + x_{n+2} + x_{n+4} \pmod 2$$

for $n \geq 0$ instead? (Compare with the LSRF above). What sequence do you get? What is the period?

Exercise 202.17.2. Compute all the possible degree 1 LSRF bit sequences. How many possibilities are there? What is the period for each of them?

Exercise 202.17.3. Compute all the possible degree 2 LSRF bit sequences. How many possibilities are there? What is the period for each of them?

Exercise 202.17.4. Compute all the possible degree 3 LSRF bit sequences. How many possibilities are there? What is the period for each of them?

Exercise 202.17.5. Write a program that generated a sequence of integers with values or 0 and 1 using the LFSR method. You want to have a `c` array as a parameter. Your generator also need to remember bits of the sequence $x_1, ...$ needed to generate the next bits. In general, for a fixed k, you want to have an array c of size k and an array x also of size k. When you call your function, the next bit is placed in the array x, shifting the bits of x so that one bit is lost. So if you want the full sequence for analysis, you need a very long array to keep the bits before it's removed from x. For instance the main program might look like this:

```
x = [x1, x2, x3, x4, x5] # the initial bits
c = [c1, c2, c3, c4, c5] # the coefs of the linear relation
bits = [x1, x2, x3, x4, x5] # the full sequence
LSRF(c, x) # append rightmost value of x to the right of bits
LSRF(c, x) # append rightmost value of x to the right of bits
etc.
```

Besides putting the new bit into `x`, it's also a good idea to return that bit as well. Then the above becomes

```
x = [x1, x2, x3, x4, x5] # the initial bits
c = [c1, c2, c3, c4, c5] # the coefs of the linear relation
bits = [x1, x2, x3, x4, x5] # the full sequence
b = LSRF(c, x); append b to the right side of bits
b = LSRF(c, x); append b to the right side of bits
etc.
```

If you like you can also write an LFSR class. Then the above becomes

```
x = [x1, x2, x3, x4, x5] # the initial bits
c = [c1, c2, c3, c4, c5] # the coefs of the linear relation
LFRS = LRFSClass(c, x)
bits = [x1, x2, x3, x4, x5] # the full sequence
b = LSRF.run(); append b to the right side of bits
b = LSRF.run(); append b to the right side of bits
etc.
```

In the above, the code works with integers 0 and 1. For scenarios where there is a huge number of bits, the bits are packed into a register (say of size 64 bits).

Exercise 202.17.6. For a given initial sequence of bits and the sequence `c`, write a function that attempts to compute the length of the period. Try lots of examples and see if you can produce cases of extremely long periods.

Index

asymmetric cipher, 290507
attack models, 290551
attack modes, 290551

Caesar cipher, 290503
chosen ciphertext attack, 290552
chosen plaintext attack, 290552
cipher, 290506, 290507
ciphertext, 290506
confusion, 290585
congruent, 290513
cycle notation, 290590

decryption, 290506
degree 5, 290612
determinant, 290581
diffusion, 290585
digrams, 290554

encryption, 290506

general linear group, 290583

Hill's cipher, 290581

identity matrix, 290582
inverse matrix, 290581
invertible, 290560, 290581

Kerckhoffs' principle, 290506
Kerckhoffs's principle, 290506
known ciphertext attack, 290551
known plaintext attack, 290551

length, 290590

monoalphabetic, 290565
multiplicative inverse, 290560

permutation, 290590
permutation cipher, 290590
permutation matrix, 290595
plaintext, 290506
polyalphabetic, 290565
private key cipher, 290507
pseudorandom, 290613
public key cipher, 290507

relative shift, 290571

security through obscurity, 290506
symmetric cipher, 290507

trigrams, 290554