Clock Arithmetic and Remainders

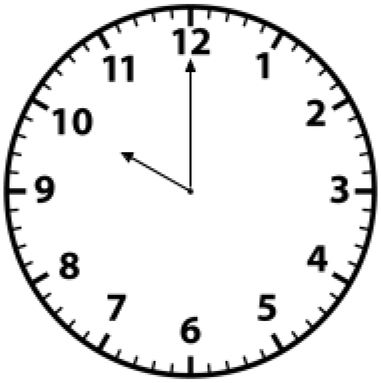
We are used to telling time in cycles of 12. Modular arithmetic is similar to how we tell the time.

Example 1

For example, if it is 10am right now, what time will it be 13 hours from now? We could say “23 o’ clock”. However, once we reach noon, we have 11 hours left. So “23 o’ clock” is the same as 11pm. The way to perform this calculation is very similar to division, except we are only interested in the *remainder*. We can write this mathematically as:

23 (mod 12) = 11

since 23/ 12 = 1 remainder 11. (We consider “12 ‘o clock” and “0 o’ clock” to be the same).



Another way to think about this is that every 12 hours, we land back on the same place on the clock. So 10 am + 13 hours is the same as cycling a full 12 hours (to get to 10pm) and then stepping forward 1 hour extra (to get to 11pm). Because time circles back on itself, we can travel by twelves and keep circling back to the same place we were originally.

Example 2

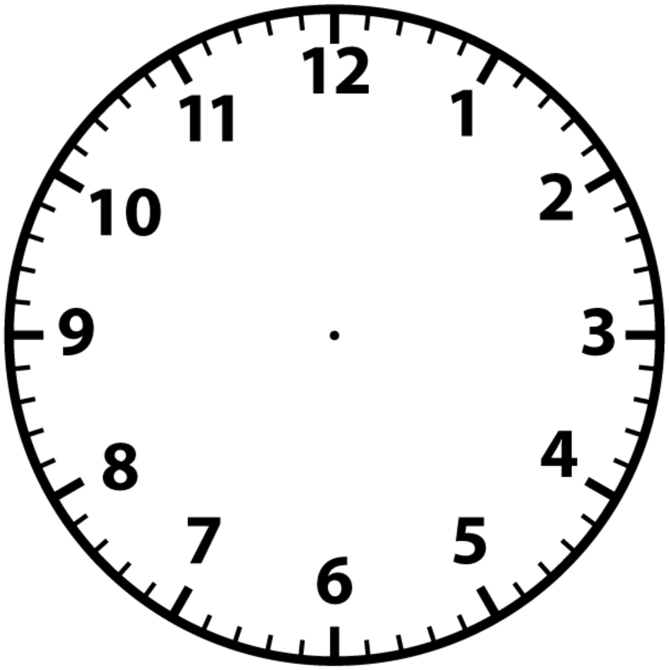
Here is one more example, but this time, we are going backwards. If it is 3 pm right now, what time was it 4 hours ago? We do not say that it was -1 pm, rather we say that it was 11 am. So we can write:

3 − 4 (mod 12) = 11

Questions

Below are some clock questions for you to try. You can draw on the empty clock picture to help you find your answers.

1. What is 11 am + 3 hours? Once we pass 12, we wrap around. So 11+3 (mod 12) = ______
2. What is 8 am + 11 hours? We can either jump by 12 and dial back by 1, or we can add 11
3. What is 6 pm - 9 hours?
4. What will be the time 49 hours from 2pm?
5. Here in North Carolina, we are 3 hours ahead of California. If it is 2 pm here, what time is it in California?
6. Sydney, Australia is 16 hours ahead of us. If it is 8pm here in Chapel Hill, what time is it in Sydney?



We can use the idea of clocks with other cycles besides 12. Let’s try this with some examples. Patterns are very important in these type of problems and can make these calculations much easier to perform, so try to find some!

(a) 5 (mod 26) =

(b) -71 (mod 70) =

(c) 101 (mod 101) =

(d) One Million (mod 2) =

(e) -52 (mod 20) =

(f) 2018 (mod 1000) =

(g) 408 (mod 7) =

(h) 36972 (mod 3) =

Making and Breaking Codes

Modular arithmetic can actually help us understand ways of both making (*encrypting*) and breaking (*decrypting*) encoded text. Imagine we have a secret message (called our *plaintext*) and want to transform it into encoded text that is hard to read (called our *ciphertext*). This way, we can try to transmit our secret while protecting against unwanted interceptors.

We will see two ways of doing this encryption, called *ciphers*. Both use substitution, where each letter in our real message always gets replaced with the same letter in our *ciphertext*. So, for example, if we replace “E” with “K”, every time “E” appears in our original message, “K” will be used in our encoded text.

Caesar Cipher

Let’s start off with the Caesar cipher. Here, we take each letter of the alphabet and shift it over by a fixed number of places, *s*.

So if we were shifting by 2, for instance, we’d replace A with the letter 2 places down, which is C. We can wrap around from the end of the alphabet back to the beginning when doing this shift. This is like “rotating” the alphabet.

For this, it is helpful to represent each letter of the alphabet with a number from 0 - 25 as in the table below. Note that there are 26 letters, but we start counting from 0, not 1:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
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

<u>Example</u>

For example, let’s use a shift of *s* = 3. If our secret message was “PIZZA”, then we would move each letter 3 places over like below:

P I Z Z A ⟶
15, 8, 25, 25, 0 ⟶ (shift by 3) ⟶
18, 11, 2, 2, 3 ⟶
S L C C D

Here, since Z is at the end of the alphabet, we cycled back over to the beginning of the alphabet to land on C.

Our final encoded text (*ciphertext*) is: “SLCCD”!

Encryption

1. Try encoding the message “YODEL” with a shift of *s* = 20. The first step has been started for you.

Plaintext	Y	O	D	E	L
Number	24	14	3	4	11
Encoded Number					
Ciphertext					

2. In the language of modular arithmetic, what are we doing when we shift a letter forwards by 20? Is there a different number we could shift backwards by?

Decryption

Let’s say we received this encoded message: LQXLXUJCN. We are told a Caesar cipher with a shift of *s*=9 was used. Starting with this code, we want to go backwards to the original message.

3. How can we undo the shift to find what each letter stands for? Is there a number we can subtract and/or add?

4. Try your idea out by cracking the message from above.

Ciphertext	L	Q	X	L	X	U	J	C	N
Encoded Number									
Number									
Plaintext									

5. Now let’s say we are doing some spying and want to decipher a message encoded with the Caesar cipher. What if we don’t know the shift but we know what 1 letter stands for (ie. X decodes into L)? Can we still decode it?

s = ______

Ciphertext	S	M	X	M	J	K
Encoded Number			23			
Number						
Plaintext			L			

6. How many possible shifts are there, including the 0-shift (no change to the original message)? Note that shifting is one-to-one, meaning that for every letter in the alphabet, we always get a unique letter after shifting.

Affine Cipher

Now, what about multiplication? It turns out there is a cipher that incorporates multiplication (mod 26). Let's first see how multiplication transforms the values of the alphabet.

Multipliers

1. Working mod 26, what happens when you use a multiply each value of the alphabet by 2? (Tip: Look for patterns)

	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
*2																										

2. Do you ever get 1 when filling out the multiplication table above?

3. In general, we want a unique way to "undo" our encryption. That is, we don't want multiple letters mapping to the same letter in our ciphertext. If all we did was multiply by 2 (mod 26) to get our ciphertext, is it possible to go backwards to get the original value uniquely? Can you find other numbers like 2 that don't give you the full alphabet after you multiply by them?

When you allow multiplication in addition to shifting, you have a more general cipher called an Affine Cipher. Here is how it goes.

Again, we will identify the letters in the alphabet with their corresponding numbers:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
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 this cipher, we now need to choose two values: a multiplier, *m*, and shift, *s*. It turns out we have to be careful about the multiplier, m, that we choose. (Similar to what you saw above, some values for *m* don't preserve the full alphabet).

The general rule to transform a letter value in our message is:

Ciphertext Value= *m* * *Plaintext Value* + *s* (mod 26)

<u>Example</u>

Let's do an example with *m*=3 and *s* = 1. (A multiplier of 3 won't produce repeats).

We take each letter's value, multiply 3, and then shift forward by 1. We are working mod 26 the whole time, so we will need to reduce our answer if it is greater than 25.

How does this work on the message "SAIL"?

S	A	I	L →
18,	0,	8,	11 → (multiply by 3 and then shift by 1)
18(3)+1,	0(3)+1,	8(3)+1,	11(3)+1 → (reduce mod 26)
3,	1,	25,	8 →
D	B	Z	I

So our *ciphertext* is "DBZI".

<u>Encryption</u>

4. Try encoding the secret message "ZERO" with a multiplier of *m* = 3 and shift of *s* = 10.

Plaintext	Z	E	R	O
Number	25	4	17	14
Encoded Number				
Ciphertext				

Inverses

In order to “undo” the Affine Cipher and uncover the original message, we’ll need the concept of an inverse.

You know inverses from regular multiplication. For example, if you know that 3 times a number is 15, you can find the original number by taking a third of 15. We say ⅓ is the inverse of 3 because when you multiply them together, ⅓ (3) = 1. They cancel eachother out.

Similarly, with multiplication (mod 26), the inverse of 3 is the number that, when multiplied by 3 and reduced mod 26, gives you 1.

Example

What is the inverse of 9 (mod 26)?

We can use trial and error. We are looking for a number from 0 to 25 that, when multiplied by 9 and reduced (mod 26), gives 1.

- If we try 9(1) (mod 26), we just get 9.
- For 9(2) (mod 26), we get 18.
- What about 9(3) (mod 26)? This gives 27 but that is the same as 1 (mod 26).

So 3 is the inverse of 9 (mod 26)!

Decryption

Now, to go backwards from an Affine cipher to the original message, we will reverse our steps. First we will need to undo the shift. Then, we will use the inverse to undo the multiplication.

5. You received this encoded message: “RBRKFP” and are told that a multiplier of m = 21 and shift of s = 1 was used. What was the original message?

Ciphertext	R	B	R	K	F	P
Encoded Numbers	17	1	17	10	5	15
Numbers						
Plaintext						

Extra

Blocks

To make codes more complex, we can use a grouping technique to break up text into blocks. For example, we can divide our original message into blocks of size 2. So now each pair of letters gets associated to one of 26 x 26 numbers.

1. Why are there 26 x 26 possible pairs?

2. How can we represent each pair of letters with a number? Try organizing the pairs like the 26 by 26 table below.

   AA, AB, AC, AD, … AZ

   BA, BB, BC, BD, … BZ

   ….

   ZA, ZB, ZC, ZD, … ZZ

Now that we know how to translate our original message into numbers, we can use the same idea earlier. We will substitute different pairs of letters with other pairs by transforming the corresponding numbers.

Citations

Adapted from:

- "Shift (Caesar) Ciphers". https://math.asu.edu/sites/default/files/shift.pdf
- "Affine Ciphers". https://math.asu.edu/sites/default/files/affine.pdf
- "LAMC Junior Circle: Modular Arithmetic and Ciphers", Preston Carroll. June 3, 2018. From https://circles.math.ucla.edu/circles/archive.shtml?year=2018
  - https://circles.math.ucla.edu/circles/lib/data/Handout-1529-1472.pdf
  - https://circles.math.ucla.edu/circles/lib/data/Handout-1530-1472.pdf
- *In Code*, Sarah and David Flannery
- Relatively Prime Stars
- https://clipart-library.com/clipart/clip-blank-analog-clock-13.htm