Mathematics and Problem Solving

Lecture 2

Modular Arithmetic

> “Mathematics is the queen of sciences and number theory is the queen of mathematics. She often condescends to render service to astronomy and other natural sciences, but in all relations she is entitled to the first rank.”
>
> Carl Friedrich Gauss

Overview

- Classes of Numbers
- Prime Factorisation
- Modular Arithmetic
  - Properties of Binary Relations
  - Properties of Binary Operators

Classes of Numbers

Numbers

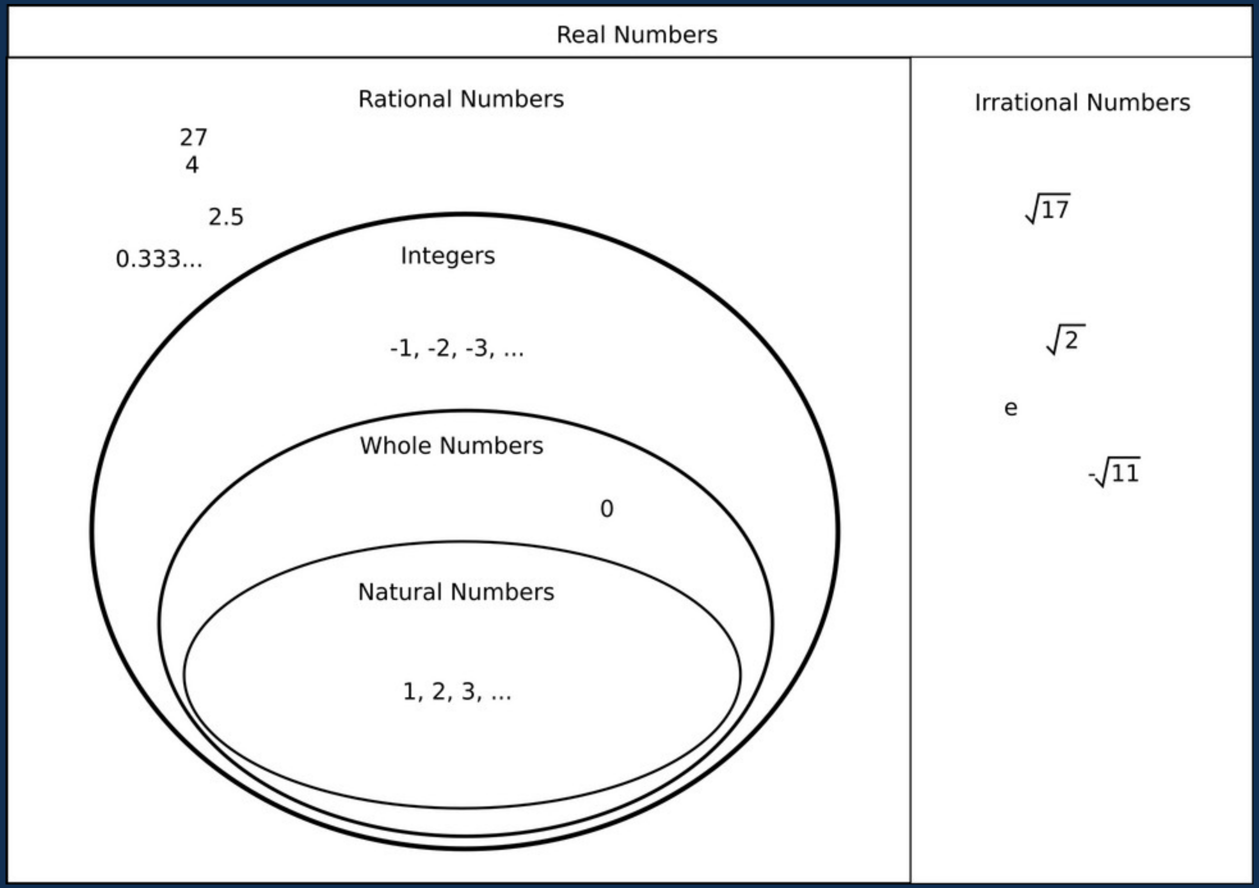
Exercise 1:

What is a number?

Classes of Numbers

- Numbers are grouped into sets called **classes**

Classes of numbers



Integers $\mathbb{Z}$

- **Integers** $\mathbb{Z}$**–** Positive or negative natural numbers or zero
  - { -2, -1, 0, 1, 2,…}
- **Naturals** $\mathbb{N}$ / $\mathbb{N}_1$/ $\mathbb{N}_+$/ $\mathbb{N}^*$ – natural counting numbers
  - { 1, 2, 3, ….}
- **Wholes** $\mathbb{N}_0$ – natural counting numbers and zero
  - { 0,1, 2, 3....}

Real Numbers $\mathbb{R}$

- Any number on a continuous 'number line'
- Two subsets
  - **Rational** $\mathbb{Q}$ – any number which can be written as a fraction
  - **Irrational** – Any decimal number which can't be written as a fraction.
    - A non-terminating and non-repeating decimal.
    - e.g. $\pi = 3.1415926...$

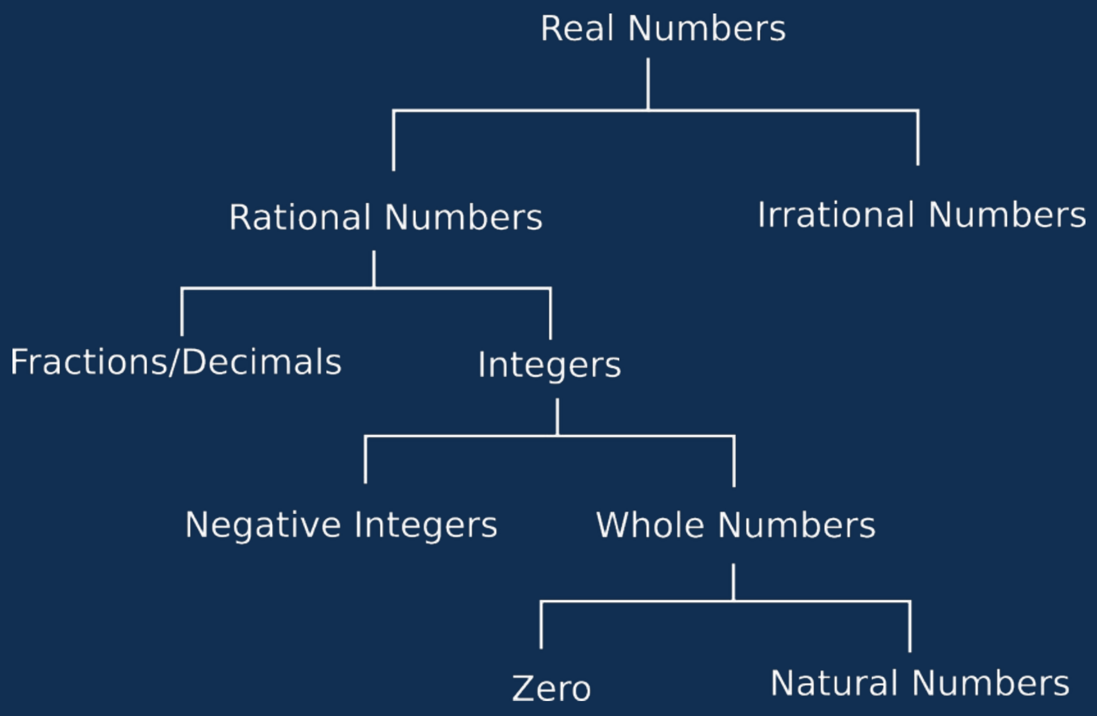
Rational Numbers $\mathbb{Q}$

- Can be expressed as a fraction (a/b).
- This set includes the integers, terminating decimals, and repeating decimals.
- Some examples:
  - 2 = 2/1
  - 3 ¼ = 13/4
  - -0.25 = -25/100
  - 1/3 = 0.33333333333333333333333

Irrational Numbers

- Cannot be expressed as a fraction of integers.
- In decimal form, they are the numbers that go on forever without a repeating pattern.
- Some examples:
  - $\sqrt{2} = 1.4142\ldots$
  - $\pi = 3.1415\ldots$
  - 45.9492…

Classes of numbers – (using a tree map)



Classes of numbers

Exercise 2:

Identify each number below as natural, whole, integer, rational, irrational, or real. More than one answer can apply.

1. 7/8
2. 0
3. -9
4. -4/5
5. π

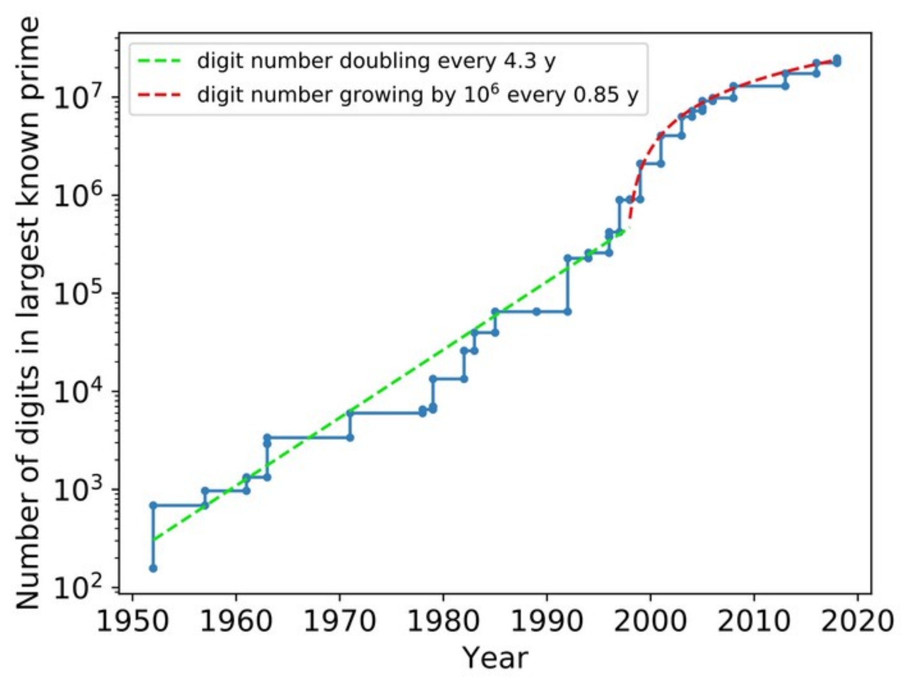
Prime Numbers

Prime Numbers

- A prime number is a whole number greater than 1 whose only factors are 1 and itself.
  - A factor is a whole numbers that can be divided evenly into another number.

Prime Numbers

- Prime numbers are of importance in mathematics
  - They are indivisible units
  - All other numbers can be composed out of prime numbers
- Large prime numbers are essential to many important cryptographic algorithms
  - Current largest is $2^{82,589,933}-1$, which is 24,862,048 digits long



Finding Primes

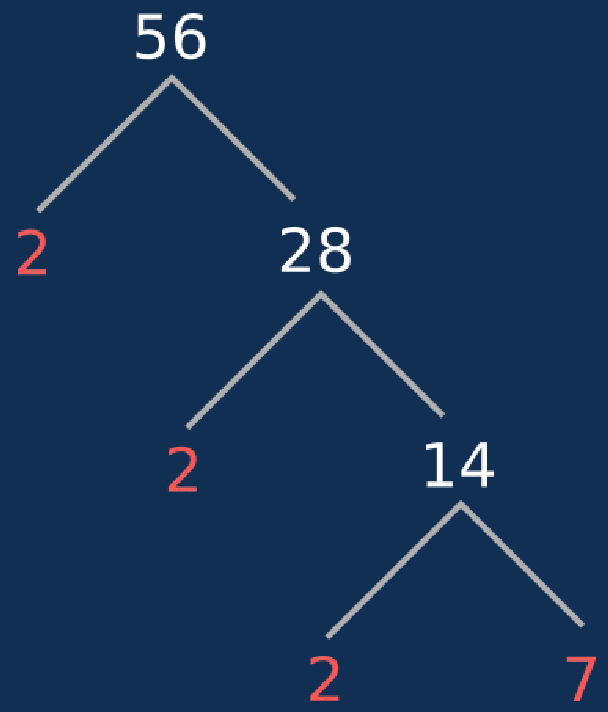
- The **Sieve of Erathosthenes** is a method for finding primes that dates from the 3rd Century BC
- Write down some integers
  - 2 3 4 5 6 7 8 9 10 11 12 13 14 15
- Start from the left: Remove all divisble by 2 (other than 2)
  - 2 3 4 5 6 7 8 9 10 11 12 13 14 15.
- Go to the next number: Remove all divisible by 3 (other than 3)
  - 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Composite Numbers

- Numbers that have more than two factors are called **composite numbers**, e.g.
  - 12 is composite as it can be factored into 2 and 6
  - 6 is composite as it can be factored into 2 and 3
- Any composite number can be expressed as the product of its prime factors
  - Every number has a unique set of prime factors
- One technique of finding the prime factors of a number is with a Factor Tree

Factor Trees

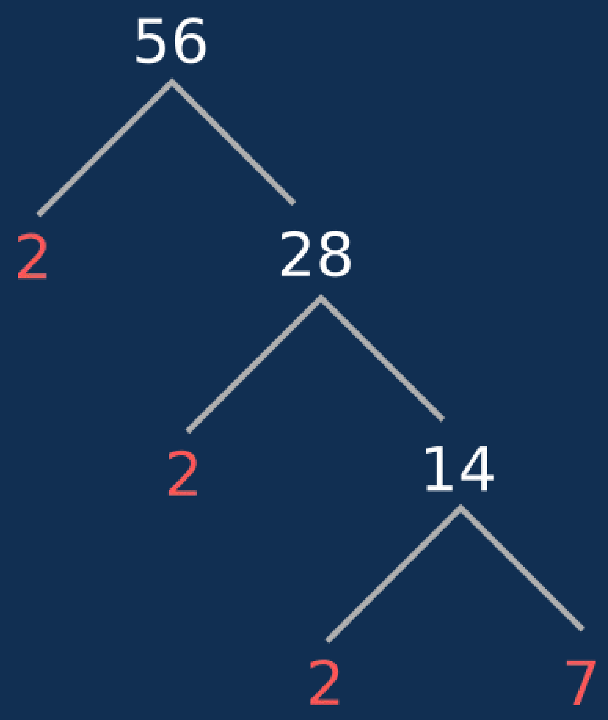
- This is a the factor tree of 56
- Red digits are prime factors of 56
- 56 = 2 x 2 x 2 x 7



Factor Trees

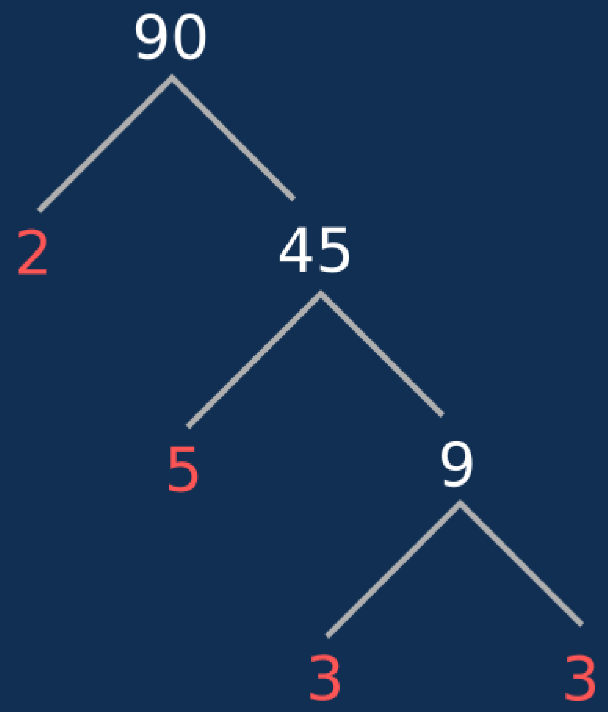
- Write 56 as a product of it's prime factors
  - List the first few primes

    2, 3, 5, 7, 11, 13, 17, 19, 23, 29
  - Find the prime that factors 56 → 2
  - Is 28 prime? No, continue.
  - Find the prime that factors 28 → 2
  - Is 14 prime? No, continue.
  - Find the prime that factors 14 → 2
  - Is 7 prime? Yes, stop



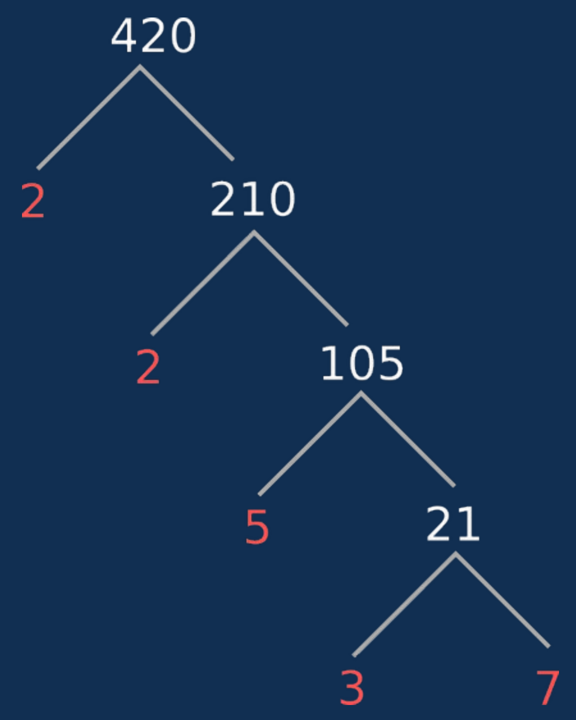
Factor Trees

- Write 90 as a product of its prime factors
- 90 = 2 x 5 x 3 x 3



Factor Trees

- Write 420 as a product of its prime factors
- 420 = 2 x 2 x 5 x 3 x 7



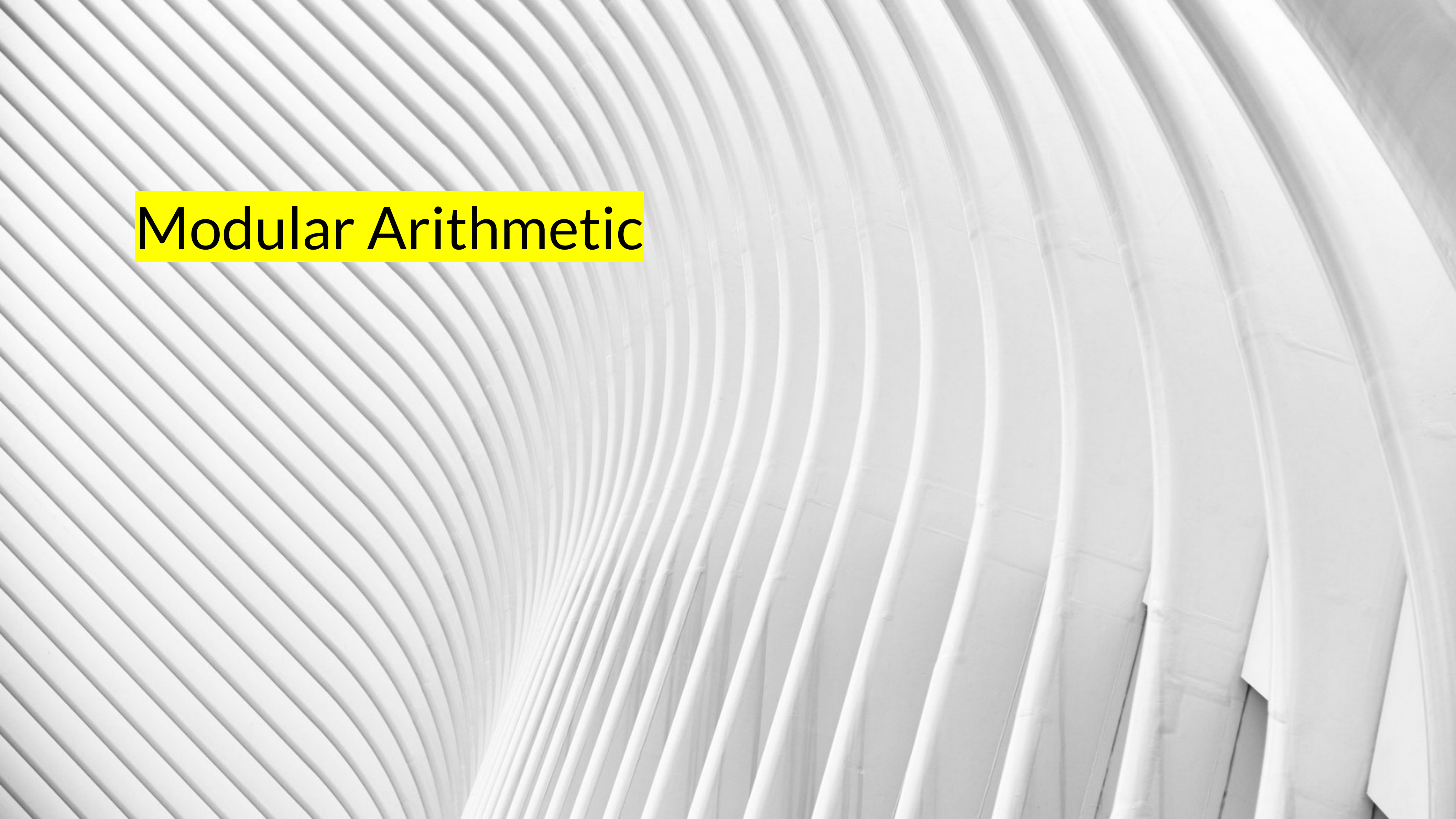
Factor Trees

Exercise 3:

Draw a factor tree and use it to write each of the following as the product of its prime factors.

1. 72
2. 80
3. 75

Modular Arithmetic



Modular Arithmetic

- Modular Arithmetic is a system of arithmetic that makes use of only a **finite set of integers**
- Analogue clocks only have 12 numbers. We are familiar with arithmetic on a clock face
  - 10 o'clock + 3 hours = 1 o'clock
  - **<u>Not 13 o'clock</u>**, 13 isn't a number in our set
  - The numbers "wrap around"

Why Modular Arithmetic?

- Why would you perform arithmetic on a finite set of integers?
  - Cyclic systems
    - Times, Dates, Numerology
  - Alphabets (finite set of letters)
    - Cryptography
  - Arithmetic algorithms on numbers in bit-representations
    - 8 bits stores only integers 0-255

What is a finite set of integers?

- With a modulus of *n* we use the numbers 0-*n*
  - We would represent a clock face as 0-11
- For example, with a modulus of 6 we use the numbers
  - {0, 1, 2, 3, 4, 5}
- All other numbers are equivalent (or **congruant**) to one of these numbers
  - E.g. if we do 5+3 = 8, that is congruant to 2

Modulo Operation

- We write the modulo operation "*a* mod *b*"
  - E.g. (5+3) mod 6 = 2
- Mod finds the remainder of a division
- In many programming languages we use the symbol %
  - Often programming languages will return fractional values for %. We will be sticking to the mathematical version

Modular Arithmetic

- When finding *n* mod *m*, remember your answer must be in the range 0→(m-1)
- To calculate the value of *n* mod *m*,
  - If *n* is positive, subtract multiples of *m*
  - If *n* is negative, add multiples of *m*
  - Until you get a number between 0 and (m-1) (inclusive)

- Examples
  - 17 mod 5 = 2
  - 20 mod 3 = 2
  - -3 mod 11 = 8
  - 25 mod 5 = 0

Modular Arithmetic

Exercise 4:

Solve the following

1. 7 mod 11
2. 11 mod 11
3. -1 mod 11
4. -11 mod 11

Congruance

Congruence

- Modular arithmetic introduces a new **binary relation** called **Congruance (≡)**
  - The following can be read “a and b are congruent modulo n”:

$$a \equiv b \pmod{n}$$

- If, when you “wrap around” *a* and *b*, you get the same value, then *a* and *b* are congruant, e.g.
  - 12 o’clock and 0 o’clock
  - 1 o’clock and 13 o’clock
  - 15 o’clock and 3 o’clock

Congruences

Congruance

Two numbers are congruant ($a \equiv b \pmod{n}$) if

$$(a \bmod n) = (b \bmod n)$$

If $a \equiv b$, the difference between a and b will be a multiple of n

$$a-b = kn, \text{ for some value of } k$$

Examples of Congruence

- $4 \equiv 9 \equiv 14 \equiv 19 \equiv -1 \equiv -6 \bmod 5$
- $73 \equiv 4 \pmod{23}$;
- $21 \equiv -9 \pmod{10}$

Reflexivity

Congruance is Reflexive:

Every number (that exists under the modulus) is congruant to itself, mod n:

$$a \equiv a \pmod{n}$$

Symmetry

Congruance is Symmetric:

If *a* is congruant to *b*, then *b* must be cogruant to *a*

$$a \equiv b \pmod{n} \text{ if } b \equiv a \pmod{n} \text{ for all a, b, and n.}$$

Transitivity

Congruance is Transitive:

If *a* is congruant to *b*, and *b* is congruant to *c*, then *a* must be congruant to *c*

If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$

Binary Relations

Exercise 5:

For the following relations, are they reflexive, symmetric, and/or transitive?

1. within 10 metres of
2. slower than
3. as tall or taller than
4. is preferred (by me) to
5. trusts

Properties of Modular Arithmetic

Properties of Modular Arithmetic

- Addition, subtaction, and multiplication work as you expect from regular arithmetic

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

Identities

Modular Identities:

$$(0 + w) \bmod n = w \bmod n$$

$$(1 \times w) \bmod n = w \bmod n$$

- An identity is an equation which is true no matter what values are chosen

Commutitivity

Commutitive Laws:

Addition and multiplication are commutative under modulus

$$(w + x) \bmod n = (x + w) \bmod n$$

$$(w \times x) \bmod n = (x \times w) \bmod n$$

Associativity

Associative Laws:

Addition and multiplication are associative under modulus

$$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$$

$$[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$$

Distributivity

Distributive Law:

Multiplication distributes over addition under modulus

$$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$$

Properties of Arithmetic

Exercise 6:

I invent a mathematical operator $\mathbb{A}$, the following are valid for all a, b, c:

- $a \mathbb{A} b = b \mathbb{A} a$
- $a \mathbb{A} (b + c) = a \mathbb{A} b + a \mathbb{A} c$
- $a \mathbb{A} (b \mathbb{A} c) \neq (a \mathbb{A} b) \mathbb{A} c$

Is $\mathbb{A}$:

1. Commutative?
2. Associative?
3. Distributive over addition?

Additive Inverse

Additive Inverse

- The **additive inverse** of a number is whatever you need to add to get 0
- In regular arithmetic, the additive inverse of a number w is -w, because

$$w + (-w) = 0$$

- However, we do not have negative numbers in modular arithmetic, the additive inverse will still exist
  - It will be positive
  - Relys on the “wrapping” effect to get to 0
  - Whatever you need to add to get to the modulus

Additive Inverse

Additive Inverse:

For each $w \in \mathbb{Z}_n$, there exists a z such that

$$w + z \equiv 0 \bmod n$$

Additive Inverse, Examples

Exercise 7:

The additive inverse is what you need to add to get 0

Find the additive inverse of the following

1. 1 (mod 2)
2. 4 (mod 10)
3. 7 (mod 11)

Multiplicative Inverse

Multiplicative Inverse

- The multiplicative inverse of *a* (written a^{-1}) is whatever you need to multiply by *a* to get 1
  - In regular arithmetic this is 1/a, e.g.
  - $4^{-1} = ¼$
- The multiplicative inverse works differently under modulus. Why?
  - Because we don't have fractions!

Multiplicative Inverse

Multiplicative Inverse:

For each $w \in \mathbb{Z}_n$, the multiplicative inverse *a* is defined where *w* and *n* are co-prime such that:

$$aw \equiv 1 \pmod{n}$$

- The multiplicative inverse is defined only if the greatest common divisor of *w* and *n* is 1 (GCD(w, n) = 1)

Multiplicative Inverse

Exercise 8:

The multiplicative inverse is the number you need to multiply by to get 1

Find the multiplicative inverse of the following:

1. 6
2. 7 (mod 13)
3. 5 (mod 7)
4. 2
5. 3 (mod 13)

Division

- You can perform division my multiplying by the multiplicative inverse, e.g.
  - $4 \div 2 = 4 \times 2^{-1}$
  - $10 \div 3 = 10 \times 3^{-1}$
- We can apply this same principle to perform division under modulus

Modular Division

- What is 5 ÷ 3 mod 11?
  - We need to multiply 5 by the inverse of 3 mod 11
  - When you multiply a number by its inverse, the answer is 1.
- The inverse of 3 mod 11 is 4 since 3*4=1 mod 11
- Thus 5 ÷ 3 mod 11 = 5 x 4 mod 11 = 9 mod 11

Modular Division

Exercise 9:

To divide under modulus, find the modular inverse of the divisor with respect to the modulus and multiply by the dividend.

Solve the following

1. $4 \div 2 \pmod 5$
2. $7 \div 5 \pmod 9$

Test for Coprimeness

- Before we can perform modular division (or the modular inverse) we must check that divisor (w) is co-prime with the modulus (n)

$$a \div w \pmod{n}$$

- This is because we need to find the modular inverse of w, which is only defined when

$$\mathrm{GCD}(w, n) = 1$$

Euclidean Algorithm

Coprime Integers

- It is sometimes interesting to know whether two integers are co-prime, meaning their greatest common divisor is 1
- Greatest Common Divisors (GCD) of two integers is the largest integer that divides both
  - If GCD(A, B) = 1 then A and B are coprime

Euclidean algorithm

- Euclidean Algorithm is a way of finding the GCD of two numbers
  - Take two numbers, a and b
  - Find b mod a
  - Repeat with
    - a =: b
    - b =: b mod a
  - Until b = 0
  - The value of a is the GCD

```
int Euclid(int a, int b)
{
    if (b == 0)
        return a;
    else
        return Euclid(b, b % a);
}
```

Representing Computation with a Table

- We can draw a table to understand its computation, e.g. for `Euclid(6,15)`

```
1. int Euclid(int a, int b) {
2.     if (b == 0)
3.         return a;
4.     else
5.         return Euclid(b, b % a); }
```

line	a	b	b=0	`b mod a`	return
1	6	15			
2			F		
5				15	
1	15	3			
2			F		
5				0	
1	3	0			
2			T		
3					3

Finding the Greatest Common Divisor

Exercise 10:

Using the Euclidean Algorithm, find

1. GCD(42, 78)

```
int Euclid(int a, int b) {
    if (b == 0)
        return a;
    else
        return Euclid(b, b % a);
}
```

Ciphers

Ciphers

- A Cipher is a way to encrypt text to make it hard for other people to read
- The most familiar cipher is the monoalphabetic substitution cypher
  - replaces every occourance of a letter with the same letter each time
- Ciphers often involve modular arithmetic
  - (Need to transform inputs to outputs within same finite alphabet)
- We'll see two examples
  - Ceasar Cipher (shift cypher)
  - Affine Cipher

Caesar Cipher

- Earliest known substitution cipher
  - Invented by Julius Caesar
  - Each letter is replaced by the letter three positions further down the alphabet.

Exercise 11:

Decode the phrase

1. KHOOR

a	b	c	d	e	f	g	h	i	j	k	l	m
D	E	F	G	H	I	J	K	L	M	N	O	P
n	o	p	q	r	s	t	u	v	w	x	y	z
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Caesar Cipher

- Defined mathematically as a pair of functions, for encryption and decryption

$$E(X) = (X + k) \bmod 26$$

$$D(X) = (X - k) \bmod 26$$

- It has a single key, *k*, which historically was k=3

Affine Cipher

- The Affine Cipher is defined by the pair of functions

$$E(X) = (aX + b) \bmod 26$$

$$D(X) = a^{-1}(X - b) \bmod 26$$

- It has two keys, *a* and *b*
- Note that to decrypt we need to calculate the modular inverse (a^{-1})
  - Because of this the value of *a* must be co-prime with the modulus 26

Public Key Cryptography

Conundrum

- If you meet a person in advance of encoding messages you can
  - Agree on a secure key
  - Or establish a One Time Pad, etc.
- How do you communicate over an insecure channel with someone with whom you’ve never met, when you haven’t already exhanged keys?

Diffie Hellman Key Exchange

- Method of exhanging keys over an insecure channel
- Commonly used
  - TLS
  - IPsec
  - SSH
  - PHP
- Works using modular arithmetic

Diffie Hellman Key Exchange

- Alice and Bob agree upon
  - modulus (p) (prime usually at least 2048 bits)
  - base (g)
- Alice and Bob choose private keys
  - Alice (a)
  - Bob (b)

41536875762873659842593824756984376582763487912837582736592873684273684728938572983759283475934875938475928475928739587249587298739587298357928759827958375298763482736857298435793487958279385792873954877239759283759247859386704598679238473782673526735476235687348693869456734568276594980638490248758096039479027945982730187439759284620950293759287049502938058920983945872094860298491283750294801937109248019358103799581093750193850791395710937597019385089103951073058710393701934701938091803984091804981093801985013984019835091835019830910791803958103951903951809358109385019840193580193840198340918093851098309180019

▲ what 2048 bits looks like as a decimal!

Diffie Hellman Key Exchange

- Alice generates a public key and sends to Bob

$$A = g^a \bmod p$$

- Bob generates a public key and sends to Alice

$$B = g^b \bmod p$$

- Public
  - modulus (p) (large prime)
  - base (g)
- Private
  - Alice's private key (a)
  - Bob's private key (b)

Diffie Hellman Key Exchange

- Alice calculates the shared secret

$$s = B^a \bmod p$$

- Bob calculates the shared secret

$$s = A^b \bmod p$$

- Public
  - modulus (p) (large prime)
  - base (g)
  - Alice's public key (A)
  - Bob's public key (B)
- Private
  - Alice's private key (a)
  - Bob's private key (b)

Diffie Hellman Key Exchange

- Alice and Bob now both know s without it ever being transmitted
- It is not practical for an attacker to calculate *s*, *a*, or *b* from the information they can intercept
  - So long as large numbers are used
- s can now be used as the key for a symmetrical encryption algorithm

- Public
  - modulus (p) (large prime)
  - base (g)
  - Alice's public key (A)
  - Bob's public key (B)
- Private
  - Alice's private key (a)
  - Bob's private key (b)
  - Shared secret (s)

Exercise 12:

Assume $p = 17$, $g = 5$. Alice generates a public key $A = 6$ using the following formula: $A = g^a \bmod p$, where A is the public key and a is a private key.

1. What is Alice's private key?
2. Devise a simple algorithm to crack a private key that would work for small values of p and g

Summary

Summary

- Numbers
  - Classes of Numbers
  - Prime Numbers
- Modular Arithmetic
  - Properties of Binary Relations
  - Properties of Binary Operators
  - Ciphers

Further Reading

- Read about Modular Arithmetic and Number Theory
  - http://www.math.umbc.edu/~campbell/NumbThy/Class/BasicNumbThy.html