Chapter 1

Elementary Number Theory

"Without mathematics, there's nothing you can do. Everything around you is mathematics. Everything around you is numbers". (S. Devi)

The topic of this first chapter is elementary number theory, that is the study of numbers and some of their basic properties.

Definition 1. Natural numbers are numbers used for counting, that is

$$1, 2, 3, 4, 5, 6, \ldots$$

Whether 0 belongs to natural numbers is often a matter of convention. We will refer to whole numbers to indicate that the zero should be included:

$$0, 1, 2, 3, 4, 5, 6, \ldots$$

The set of natural numbers is often denoted by $\mathbb{N}$:

$$\mathbb{N} = \{1, 2, 3, 4, 5, 6, \ldots\}.$$

Definition 2. Integer numbers are natural numbers, with their opposites (natural numbers with a negative sign), and zero, that is

$$\ldots, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, \ldots$$

The set of integer numbers is often denoted by $\mathbb{Z}$:

$$\mathbb{Z} = \{\ldots, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, \ldots\}.$$

Integer and Real Numbers

- **Natural numbers** (*N*) are the counting numbers:

  1, 2, 3, ...

  – Sometimes 0 is also included (**whole** numbers)
- **Integers** (*Z*) are the natural numbers, including zero, and their negatives:

  … -3, -2, -1, 0, 1, 2, 3, …
- **Real numbers** (*R*) are any value on the continuous line.
  – for example: 0.31, -4, π , 2

Ir(rational) Numbers

- **Rational numbers** (*Q*) are real numbers which can be represented in the form a/b, where a and b are integers
  – for example: 3/7, 0.999 = 999/1000, …
- **Irrational numbers** (*I*) are real numbers which can NOT be represented in the form a/b for any integers a & b
  – for example: π, *e*, $2^{1/2}$

Real numbers are difficult to define formally. For us, we will say that real numbers are values on the continuous real line. Real numbers include for example $\sqrt{2}$, or π, but also -3 or 0.5.

Definition 3. Rational numbers are real numbers that can be written as the ratio of two integer numbers. The set $\mathbb{Q}$ of rational numbers is formally defined as

$$\mathbb{Q} = \left\{ \frac{a}{b},\ a, b \in \mathbb{Z},\ b \neq 0 \right\}.$$

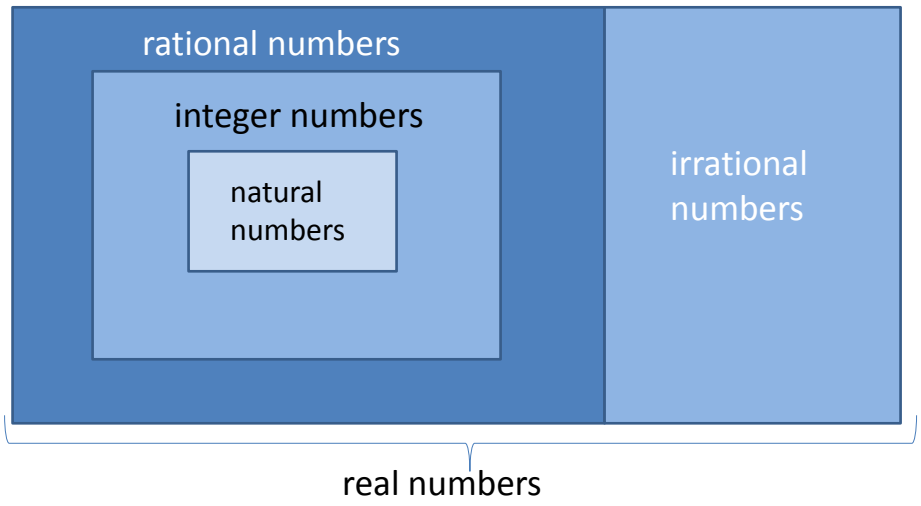
Definition 4. Irrational numbers are real numbers that **cannot** be written as the ratio of two integer numbers.

It is easier to show that a number is rational than to show that it is irrational. Indeed, to show that a number is rational, it is enough to find two integers a and b such that a/b is your number. For example, 0.5 is rational, because if you take $a = 1$ and $b = 2$, then $a/b = 1/2 = 0.5$. There are other choices of a and b. For example, you could have taken $a = 2$ and $b = 4$, then $a/b = 2/4 = 1/2 = 0.5$. However, to show that a number is irrational, you need to show that no matter which choice of integers a and b you take, then a/b will never be your number (see Exercise 3 for an example).

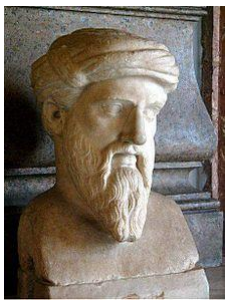
We already saw 5 types of numbers: natural, integer, real, rational, and irrational. To remember them, it may be useful to think about how these types of numbers are connected.

- Real numbers form the largest set of numbers that we will encounter for many chapters. All 4 other sets of numbers are included in $\mathbb{R}$.
- Both rational numbers ($\mathbb{Q}$) and irrational numbers are real numbers. These two sets are disjoint, that is there is no number which is both rational and irrational! Either a number is a ratio of two integers, or it is not. One way to help in remembering the term "rational" is to think of a "ratio" of two integers.
- The set $\mathbb{Z}$ of integer numbers is included in $\mathbb{Q}$. Indeed, take any integer number a (positive, negative, or 0), then $a = a/1$, thus a is always the ratio of two integers.
- The set $\mathbb{N}$ is included in $\mathbb{Z}$.

Numbers

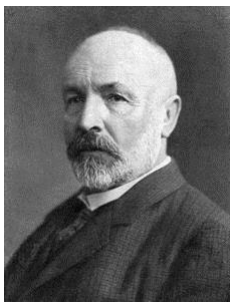


Numbers and Mathematicians



Pythagoras
(Πυθαγόρας)
c. 570 BC-c. 495 BC

Hippasus (Ἵππασος)
of Metapontum
5th century BC



Georg Ferdinand
Ludwig Philipp Cantor
1845 -1918

Pictures from Wikipedia

Mathematicians have been continuously studying numbers. The notion of irrationality is as old as 5th century BC. The school of Pythagoreans (Greek philosophers who followed the mathematician and philosopher Pythagoras) believed that all numbers were rationals. The legend has it that Hippasus, who is believed to have discovered the irrationality of some numbers, may have been drowned, either to be prevented to report this finding, or to punish him for it! From the Pythagoreans time, it took more than 20 centuries to mathematicians to come up with a formal definition for real numbers, which is due to the mathematician Cantor. Numbers are still studied by mathematicians nowadays, in a branch of mathematics called number theory.

Numbers have also been of great interest for computer scientists. While it is fairly easy for a computer to represent an integer number (not "too big") in its memory, it is much more complicated for real numbers in general, and in fact, many real numbers need to be approximated, leading to potential numerical problems. Rational numbers are much nicer, exactly because they can be represented by a pair of integers. Some programming languages (such as C), will require you to handle *types*, and thus to identify which numbers you are working with.

What we call now Euclidean division was discovered after irrational numbers.

Definition 5. The Euclidean division states that for any positive integer n and any integer m, there exist unique integers q (called quotient) and r (called remainder) such that

$$m = qn + r,\ 0 \leq r < n.$$

Pay attention to the condition on r, it is important. It tells that r is strictly smaller than the number n with which we divide m. When the remainder $r = 0$, then we say that n divides m, also written

$$n \mid m.$$

Alternatively, we may say that n is divisible by m.

Numbers and Computer Science

```
gap>
gap>
gap> 10/3;
10/3
gap> 10.0/3;
3.33333
gap>
```

- Real numbers need approximation!
- Rational numbers = pair of integers
- Type of numbers (e.g. in C)

frac.c

```c
#include <stdio.h>

void main()
{
    float a;
    a=10/3;
    printf("%f\n",a);
    a=(float)10/3;
    printf("%f\n",a);
}
```

```
frederique@frederique-desktop:~$ gcc frac.c -o frac
frederique@frederique-desktop:~$ ./frac
3.000000
3.333333
frederique@frederique-desktop:~$
```

Euclidean Division

- Take any integer n, $n>0$, and any integer m. There exist unique integers q and r such that

$$m = qn+r,\ 0 \le r < n.$$

- q = quotient, r = remainder
- When $r=0$, then
  - n divides m, or
  - m is divisible by n.
  - Notation: $n \mid m$

Euclide (Εὐκλείδης) 300 BC

Picture from Wikipedia

Example 1.

If $n = 7$ and $m = 17$, then we want to write $17 = 7q + r$, with $0 \leq r < 7$. We get

$$17 = 7 \cdot 2 + 3,\ q = 2,\ r = 3.$$

If $n = 7$ and $m = 35$, then we want to write $35 = 7q + r$, with $0 \leq r < 7$. We get

$$35 = 7 \cdot 5 + 0,\ q = 5,\ r = 0.$$

Thus $7 \mid 35$, in words, 7 divides 35.

What happens if m is a negative number? Here is an example. If $n = 7$ and $m = -5$, then we want to write $-5 = 7q + r$, with $0 \leq r < 7$. We get

$$-5 = 7(-1) + 3,\ q = -1,\ r = 2.$$

Thanks to the notion of divisibility, we may define two new types of numbers.

Definition 6. A prime number, often denoted by p, is a natural number divisible by exactly two factors: 1 and itself.

We thus consider that 1 is not a prime number, because 1 has only one divisor, namely 1. The first prime numbers are

$$2, 3, 5, 7, \ldots.$$

But 9 is not a prime number, because $1 \mid 9$, $3 \mid 9$ and $9 \mid 9$, thus there are 3 factors, 1, 3, and 9 which divide 9.

Definition 7. An even number is an integer number divisible by 2. An odd number is an integer number which is not divisible by 2.

If a is an even number, we may formally write it as $a = 2a'$ for a' some integer, to emphasize that 2 appears in its factorization.

For example, 4 is even. Among the prime numbers, only 2 is even! (see Exercise 1). Then 9 is odd, because we just saw that its factors are 1,3 and 9, 2 is not a factor of 9.

Euclidean Division: Examples

- Take any integer n, $n>0$, and any integer m. There exist unique integers q and r such that

$$m = qn+r,\ 0 \leq r < n.$$

- E.g. $n=7$,
  - If $m = 17$, $q = 2$, $r = 3$
  - If $m = 35$, $q = 5$, $r = 0$
  - If $m = -5$, $q = -1$, $r = 2$

Prime Numbers, Even Numbers

- **Prime numbers** are natural numbers p, which have only two factors: p and 1, i.e., *not divisible* by any other integer
  - For it to have two factors, it has to be larger than 1
  - 2, 3, 5, 7, 11, 13, …
- **Even numbers** are integers divisible by 2. **Odd numbers** are integers not divisible by 2.

The notion of divisibility is useful to define a new notion, that of *modulo*.

Definition 8. For a positive integer n, two integers a and b are said to be congruent $\pmod{n}$ if $a-b$ is an integer multiple of n. We use the notation:

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

The notation $\equiv$ emphasizes that we do not have a "normal" equality here.

However, $a \equiv b \pmod{n}$ can be turned into an equality as we see next. We write that $a-b$ is an integer multiple of n as $a-b=qn$ for some integer q, then

$$a = qn + b.$$

Let us see examples, where we will further comment on the link with the Euclidean division introduced above.

Example 2.

Take $a=-8$, which we want to compute modulo $n=5$. Then the Euclidean division tells us that we can write $a=-8=q5+b$ with a b such that $0 \leq b < 5$. We get

$$-8 = 5\dot{(}-2) + 2,\ b = 2.$$

Thus $-8 \equiv 2 \pmod{5}$. Now in the definition of modulo n, we are allowed to take b bigger, so we could write

$$-8 = 5\dot{(}-3) + 7,\ b = 7$$

and further obtain that $-8 \equiv 2 \equiv 7 \pmod{5}$. Unlike for the Euclidean division, infinitely many choices of b are possible.

Suppose that we want next to compute $17 \mod 5$. "Informally", we are allowed to add and subtract multiples of 5. Thus if one removes $3 \cdot 5$ to 17, we get 2, if one removes $1 \cdot 5$ to 17, we get 12, and if one adds $1 \cdot 5$, this gives 22. To summarize

$$17 \equiv 2 \equiv 12 \equiv 22 \pmod{5}.$$

Given an integer a, there are many (infinitely many) b such that $a \equiv b \pmod{n}$. However, using the Euclidean division, there is only one b between 0 and $n-1$. This is why we can represent integers $\pmod{n}$ by the set

$$\{0, 1, \ldots, n-1\}.$$

Modulo n

- For a positive integer n, two integers a and b are said to be congruent modulo n, if $a - b$ is an integer multiple of n. We write

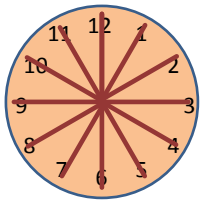
$$a \equiv b \pmod{n}$$

- If $a \equiv b \pmod{n}$, then $a\text{-}b = qn$ and $a=qn+b$.

Modulo n: Examples

$$a \equiv b \pmod{n} \leftrightarrow a=qn+b$$

- $-8 \equiv 2 \equiv 7 \pmod{5}$
- $17 \equiv 2 \equiv 12 \pmod{5}$
- Likewise $17 \equiv 22 \pmod{5}$



Now that we have a new set of numbers, that of integers modulo n, one may wonder whether it is possible to compute with these numbers. And the answer is yes! It is possible to add two numbers modulo n, and to multiply them.

Let us try out with addition. Suppose that we have $17 \pmod 5$ and $-8 \pmod 5$, and we want to add them up:

$$(7 \pmod 5) + (-8 \pmod 5) = ?$$

Shall we first compute $7 \pmod 5 = 2$, and $-8 \pmod 5 = 2$, and then add them to find that it is $4 \pmod 5$? Or should we first compute $7+(-8) = -1$, and then compute $-1 \pmod 5$? It turns out that it does not matter (see Exercise 4), neither does it matter for multiplication, which is why computations are possible: they are consistent.

We can summarize how to compute modulo n using an addition table or a multiplication table. Let us try out with integers modulo 2, and addition:

+	0	1
0	0	1
1	1	0

.

We notice that this table is "almost normal", but for the fact that $1+1 = 0$! Similarly, we get for integers modulo 2 and multiplication:

*	0	1
0	0	0
1	0	1

.

Modular Arithmetic

$a \equiv b \pmod{n} \leftrightarrow a=qn+b$

- Integers mod n can be represented as elements between 0 and $n-1$: {0,1,2,…,$n-1$}
- Define addition mod n and multiplication mod n:

  $(a \bmod n) + (b \bmod n) \equiv (a+b) \bmod n$

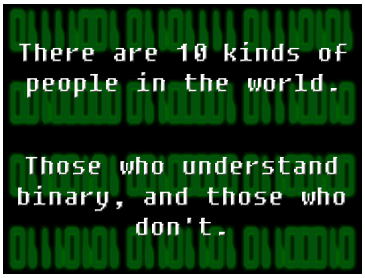
  $(a \bmod n) * (b \bmod n) \equiv (a*b) \bmod n$
- Examples
  - (17 mod 5) + (-8 mod 5) ≡ 4 mod 5
  - (12 mod 5) * (-3 mod 5) ≡ 4 mod 5

Integers mod 2

- Bits are integer modulo 2

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1



We have seen many sets of numbers, all of them coming with "a way to compute", that is to perform either addition, or multiplication, sometimes division. Consider more generally any set S, with any operator Δ.

Definition 9. Consider a set S with an operator Δ. Then S is closed under Δ if the result of the operation Δ on any two elements of S results in an element of S. This property is known as closure.

Example 3.

- If $S = \mathbb{R}$, the set of real numbers, and $\Delta = +$ is the addition, then the addition of two real numbers results in another real number. Thus S is closed under addition.
- If $S = \mathbb{Z}$, the set of integer numbers, and Δ is the division, then S is not closed under Δ. Indeed, divide 1 by 2, both 1 and 2 are integer numbers, but $1/2$ is not!
- The above definition can be applied to integers modulo n as well. Take S to be the set of integers modulo n, that is $S = \{0, 1, \ldots, n-1\}$, and Δ is the addition modulo n, then S is closed under Δ.

Why do we care about the closure property? Well, we usually care, because we like structure! A set as such is very generic, once we start adding an operator on it, and this set turns out to be closed under this operator, then we are gaining more structure! This becomes handy for example when one wants to write an algorithm on a set.

Operator Closure

- Consider a set *S* with an operator Δ. Then S is closed under Δ if the result of the operation Δ on any two elements of S results in an element of S.
  - This is known as the ***closure*** property.
- Examples:
  - S=R={real numbers} is closed under Δ=+ (and Δ=*).
  - S=Z={integer numbers} is not closed under Δ=division.
  - S={integers mod *n*} is closed under Δ=addition mod *n*.

Summary

- Recognize different types of numbers (natural, integer, real, rational, irrational, prime, even, modulo n)
- Decide whether these sets of numbers are closed under a given operator.

Calvin & Hobbes belong to Bill Watterson

Exercises for Chapter 1

Exercise 1. Show that 2 is the only prime number which is even.

Exercise 2. Show that if n^2 is even, then n is even, for n an integer.

Exercise 3. The goal of this exercise is to show that $\sqrt{2}$ is irrational. We provide a step by step way of doing so.

1. Suppose by contradiction that $\sqrt{2}$ is rational, that is $\sqrt{2} = \frac{m}{n}$, for m and n integers with no common factor. Show that m has to be even, that is $m = 2k$.

2. Compute m^2, and deduce that n has to be even too, a contradiction.

Exercise 4. This exercise is optional, it requires to write things quite formally. Show the following two properties of integers modulo n:

1. $(a \mod n) + (b \mod n) \equiv (a + b) \mod n$.

2. $(a \mod n)(b \mod n) \equiv (a \cdot b) \mod n$.

Exercise 5. Compute the addition table and the multiplication tables for integers modulo 4.

Exercise 6. Show that $\frac{p(p+1)}{2} \equiv 0 \pmod{p}$ for p an odd prime.

Exercise 7. Consider the following sets S, with respective operator Δ.

- Let S be the set of rational numbers, and Δ be the multiplication. Is S closed under Δ? Justify your answer.

- Let S be the set of natural numbers, and Δ be the subtraction. Is S closed under Δ? Justify your answer.

- Let S be the set of irrational numbers, and Δ be the addition. Is S closed under Δ? Justify your answer.

Examples for Chapter 1

A first application of integers modulo 2 is counting in binary. In order to represent a natural number, we may write it in terms of power of 2:

$$2^0 = 1,\ 2^1 = 2,\ 2^2 = 4,\ 2^3 = 8,\ 2^4 = 16, \ldots$$

Suppose that a number n may be written as

$$n = n_0 2^0 + n_1 2^1 + n_2 2^2 + \cdots + n_r 2^r$$

for some positive integer r. Then we may write n in binary as

$$n = (n_r\ n_{r-1}\ \cdots n_1\ n_1\ n_0).$$

Now all the n_i, $i = 0, \ldots, n_r$ are either 0 or 1!

Example 4. Take $n = 18$. Then

$$\begin{aligned} 18 &= 16 + 2 \\ &= 2^4 + 2^1 \end{aligned}$$

and

$$18 = (10010).$$

Our main application for binary arithmetic will be coming from storage. In order to explain it, we need the notion of binary vector. A vector in general is an array containing numbers (this array is either horizontal or vertical). For example, here are a row (horizontal) and a column (vertical) vector:

$$(a_1, a_2, \ldots, a_n), \quad \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}.$$

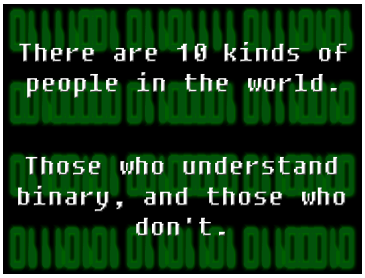
A binary vector contains only 0 and 1. Its length is the number of 0 and 1 it contains. For example $(1, 0, 0, 1)$ is a binary vector of length 4.

Counting in Binary

$10010_2 = 0*2^0 + 1*2^1 + 0*2^2 + 0*2^3 + 1*2^4$

$= 18_{10}$

$= 8.10^0 + 1.10^1$



Binary Vectors

- A vector is a row (or column) array containing numbers.
- (1,0,0,1) is a binary row vector (of length 4)
- One can add two binary vectors component wise (vector addition).

One may add two vectors componentwise:

$$(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n).$$

Example 5. We have

$$(1, 0, 0, 1) + (0, 0, 1, 1) = (1, 0, 1, 0)$$

since $1 + 0 = 1$, $0 + 0 = 0$, $0 + 1 = 1$ and $1 + 1 = 0$.

We have seen in this chapter the notion of closure of a set S under an operator Δ (see Definition 9). Let us see how to apply it here.

Example 6. Let S be the set of binary vectors of length n, and Δ be the vector addition. Is S closed under Δ? To check this, we need to make sure that the addition of any two binary vectors will result in an element of S, that is a binary vector of length n. Suppose then that $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ are both in S. Then

$$(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n),$$

and since $a_i + b_i$ is either 0 or 1, then $(a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$ is in S, thus S is closed under Δ.

Note that binary vector addition is different from what we do to count in binary.

Example 7. We continue Example 4. We saw that

$$18 = (10010) = (n_4, n_3, n_2, n_1, n_0).$$

How do we count $18 + 18$ in binary, using that $18 = (10010)$? We start from $n_0 = 0$, and thus $n_0 + n_0 = 0$. Thus

$$18 + 18 = (*, 0).$$

We look at n_1 next. We have $n_1 + n_1 = 1 + 1 = 0$, however here, it means that $n_1 2^1 + n_1 2^1 = 2^1(n_1 + n_1)$ thus we do get a 0 in the second position from the right, but there is a 1 which is carried over!

$$18 + 18 = (*, 0, 0).$$

Binary Vectors: Example

- For example (1,0,0,1)+(0,0,1,1)=(1,0,1,0)
- S=set of binary vectors of length n, Δ=vector addition. Is S closed under Δ?
- Different from counting in binary!

Binary in the Real World

- Storage of data across multiple hard disks
- Data is in binary format.
- Data needs to be stored so as to tolerate disk failures.

Image belongs to Microsoft

To compute the next term, we have $n_2 + n_2 = 0$, but there is the 1 carried over, so we get

$$18 + 18 = (*, 1, 0, 0).$$

Then $n_3 + n_3 = 0$, and $n_4 + n_4 = 0$ with again a 1 carried over, to finally get

$$18 + 18 = (1, 0, 0, 1, 0, 0) = 1 \cdot 2^5 + 1 \cdot 2^2 = 32 + 4$$

as it should be.

Now that we have seen how to perform vector additions (which is different from adding two numbers written in binary), let us go back to our storage application.

The engineering problem that we are facing is that of storing data across several hard disks, and in fact, data is stored in binary (so you can imagine the stored data as a binary vector). Now if your data is stored only on one disk, if this disk suffers from a failure, you are likely to lose your data. To prevent this to happen, typically you will store your data on two hard disks, each disk will contain one copy.

Example 8. Suppose you want to store 200GB of data, and the shop is selling disks of 100 GB each. Then you can buy 4 disks, store half of your data (let us call it D_1) on disk 1, the other half (say D_2) on disk 2, then copy the content of disk 1 to disk 3, and the content of disk 2 to disk 4. We get thus the following data allocation:

$$\text{disk } 1 : D_1,\ \text{disk } 2 : D_2,\ \text{disk } 3 : D_1,\ \text{disk } 4 : D_2.$$

- **Good**: what is good now is that even if any of the 4 disks fails, your data is still safe. Note though that only one disk failure is ok, two failures is not, because if it happens to both disk 1 and disk 3 for example, then your data is lost.

- **Bad**: what is bad is that you did pay for 4 disks, while you only need 2 for the actual data...but then with only 2 disks, any failure will mean that your data is lost...

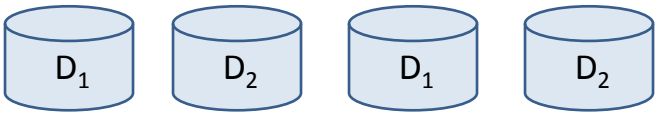
So there is a trade-off here: either you pay for 2 disks and tolerate no failure, or you pay for 4, but still can tolerate one failure. Is there any way to get a better trade-off?

Since 2 disks give no protection against failures, we need at least 3 disks. The question is then, can we have only 3 disks, yet tolerate any one of the 3 disks to fail?

Example (I)

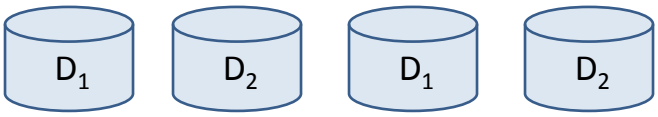
- Suppose you want to store 200GB of (binary) data
- Option 1: buy 4 disks of 100 GB each, store 2 copies of your data.

$D = (D_1, D_2)$



Example (II)

$D = (D_1, D_2)$



- Good thing: if one hard disk fails, your data is safe.
- Bad thing: you paid for 4 hard disks instead of 2.
- Can we think of a better solution?

If you put your data D_1 and D_2 in each of the first 2 disks

$$\text{disk } 1 : D_1, \ \text{disk } 2 : D_2, \text{disk } 3 :?$$

then putting either D_1 or D_2 in the 3rd disk will not help. So what else could we do? We could use binary operations on D_1 and D_2, namely compute the sum mod 2 (also called XOR) of D_1 and D_2:

$$\text{disk } 1 : D_1, \ \text{disk } 2 : D_2, \ \text{disk } 3 : D_1 + D_2.$$

Let us make sure we understand what this means, by first looking at 3 bits, $a, b, a + b$, that is $a, b, a + b$ are three integers mod 2. Now you have three cases:

1. suppose you lose a: then you have b and $a+b$, compute $b+(a+b) = a$.
2. suppose you lose b: then you have a and $a+b$, compute $a+(a+b) = b$.
3. suppose you lose $a + b$: then you have a and b.

In all cases, both a and b can be recovered from one missing bit.

We thus do the same thing with disks (drives), except that now, disks contain *binary vectors* instead of just one bit. Let us do an example, with $D_1 = (0, 1, 1, 0, 1, 1, 0, 1)$, $D_2 = (1, 1, 0, 1, 0, 1, 0, 0)$:

$$\text{disk } 1 : D_1, \ \text{disk } 2 : D_2, \ \text{disk } 3 : D_1 + D_2 =?.$$

We compute the binary vector addition of D_1 and D_2:

$$D_3 = D_1 + D_2 = (1, 0, 1, 1, 1, 0, 0, 1).$$

This strategy is thus better than the previous one: you are paying for one disk less (three hard disks instead of 4), and you still can tolerate any one disk failure.

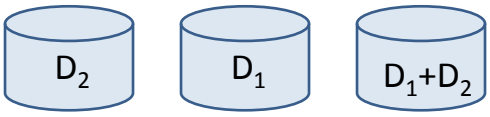
Example (III)

- Suppose a and b are bits, and take a, b, a+b

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	0

- Do the same thing with disks

$D = (D_1, D_2)$



Example (IV): Parity (RAID)

- Binary vector addition:

Drive 1: **01101101**
Drive 2: **11010100**

01101101
XOR 11010100
_____
10111001 ← Store in Drive 3

You can still loose one disk, but paid for only 3.

Example 9. Let us try another storage example. You have 150 GB of data to store, and you are buying hard disks, each of 50GB storage capacity each. But unlike in the previous example where you were happy with one disk failure protection, this time you would like protection against any 2 failures!

Let us start, you have 150 GB of data, and disks of size 50 GB, so you split your data into 3 equal parts D_1, D_2, D_3:

$$\text{disk } 1 : D_1,\ \text{disk } 2 : D_2,\ \text{disk } 3 : D_3.$$

Here is now a *key observation*: to tolerate any 2 failures, each D_i, $i = 1, 2, 3$ must appear at least 3 times! (if any appears only twice, when the corresponding 2 disks fail, then the data is lost...)

Can we manage something clever with only 4 disks?

$$\text{disk } 1 : D_1,\ \text{disk } 2 : D_2,\ \text{disk } 3 : D_3,\ \text{disk } 4 :?.$$

It is not possible to add only one piece of data in disk 4, and have D_1, D_2, D_3 present at least 3 times. Let us try with 5 disks:

$$\text{disk } 1 : D_1,\ \text{disk } 2 : D_2,\ \text{disk } 3 : D_3,\ \text{disk } 4 :?,\ \text{disk } 5 :?.$$

The only way to have D_1, D_2, D_3 present at least 3 times is to put $D_1+D_2+D_3$ both in disk 4 and disk 5. Let us see whether you are protected against 2 failures. If D_1 and D_2 fail, you are left with

$$D_3, D_1 + D_2 + D_3.$$

It is only possible to recover $D_1 + D_2$...so we have to move to 6 disks:

$$\text{disk } 1 : D_1,\ \text{disk } 2 : D_2,\ \text{disk } 3 : D_3,\ \text{disk } 4 :?,\ \text{disk } 5 :?,\ \text{disk } 6 :?.$$

Now we have space to put D_1, D_2, D_3 such that they are each present at least 3 times. We are the possible choices of data to be put in your empty disks?

$$D_1 + D_2,\ D_1 + D_3,\ D_2 + D_3,\ D_1 + D_2 + D_3.$$

Let us try:

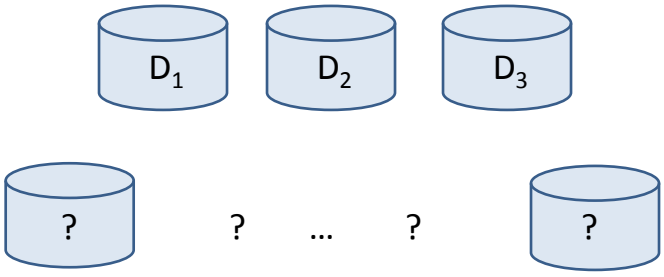
$$\begin{array}{lll} \text{disk } 1 : D_1, & \text{disk } 2 : D_2, & \text{disk } 3 : D_3, \\ \text{disk } 4 : D_1 + D_2, & \text{disk } 5 : D_1 + D_3, & \text{disk } 6 : D_2 + D_3. \end{array}$$

Another Example (I)

- You want to store 150 GB data.
- Now you are buying storage devices, each of 50GB capacity.
- This time, even if any arbitrary two devices fail, you still want to recover all your data!

Another Example (II)

$D = (D_1, D_2, D_3)$



We can achieve what we want with a total of 9 disks. Can we do better?

Let us first look at the case where two of the D_i are gone:

1. If we lose D_1, D_2: we are left with only D_3, but D_3 with disks 5 and 6 works.

2. If we lose D_1, D_3: we are left with only D_2, but D_2 with disks 4 and 6 works.

3. If we lose D_2, D_3: we are left with only D_1, but D_1 with disks 5 and 5 works.

We recall the data allocation for convenience:

$$\begin{array}{lll} \text{disk } 1 : D_1, & \text{disk } 2 : D_2, & \text{disk } 3 : D_3, \\ \text{disk } 4 : D_1 + D_2, & \text{disk } 5 : D_1 + D_3, & \text{disk } 6 : D_2 + D_3. \end{array}$$

We look next at the case where one of the D_i is gone, and the other one is either disk 4, 5, or 6:

1. If we lose D_1: either disk 4 or 5 can be used.

2. If we lose D_2: either disk 4 or 6 can be used.

3. If we lose D_3: either disk 5 or 5 can be used.

So this strategy works! It needs 6 hard disks, which is less costly than making 3 copies of the data, which would have needed 9 disks...

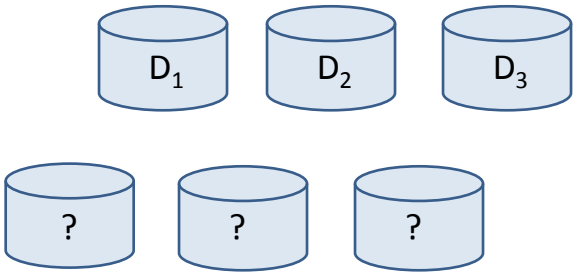
It is not the only strategy that works, you may come up with your solution, which is different than this one, and also works!

If you are wondering whether this way of thinking can be used for other examples, yes it can, as long as the numbers of disks are not too big, after which it becomes more tiresome...

If you are wondering whether there is a trick to make things faster, yes, there is one, it is called *the Hamming distance*, but it is beyond the scope of this course, so feel free to check it out if you are curious, but the notion of Hamming distance **is not needed** for this course.

Another Example (III)

$D = (D_1, D_2, D_3)$



D_1+D_2
D_1+D_3
D_2+D_3
$D_1+D_2+D_3$

To tolerate two failures, we need each D_i to be present at least 3 times.

Summary

- Modulo n
- Binary arithmetic
  - Binary vectors
  - Counting in binary
- Applications of binary arithmetic
  - storage