Advanced information, computation, communication I
Prof. Aberer Karl — EPFL

Notes by Joachim Favre

Computer science bachelor — Semester 1
Autumn 2021

I made this document for my own use, but I thought that typed notes might be of interest to others. So, I shared it (with you, if you are reading this!); since it did not cost me anything. I just ask you to keep in mind that there are mistakes, it is impossible not to make any. If you find some, please feel free to share them with me (grammatical and vocabulary errors are of course also welcome). You can contact me at the following e-mail address:

`joachim.favre@epfl.ch`

If you did not get this document through my GitHub repository, then you may be interested by the fact that I have one on which I put my typed notes. Here is the link:

`https://github.com/JoachimFavre/EPFLNotesIN`

Please note that the content does not belong to me. I have made some structural changes, reworded some parts, and added some personal notes; but the wording and explanations come mainly from the Professor, and from the book on which they based their course.

I think it is worth mentioning that in order to get these notes typed up, I took my notes in LaTeXduring the course, and then made some corrections. I do not think typing handwritten notes is doable in terms of the amount of work. To take notes in LaTeX, I took my inspiration from the following link, written by Gilles Castel. If you want more details, feel free to contact me at my e-mail address, mentioned hereinabove.

`https://castel.dev/post/lecture-notes-1/`

I would also like to specify that the words "trivial" and "simple" do not have, in this course, the definition you find in a dictionary. We are at EPFL, nothing we do is trivial. Something trivial is something that a random person in the street would be able to do. In our context, understand these words more as "simpler than the rest". Also, it is okay if you take a while to understand something that is said to be trivial.

Since you are reading this, I will take give you a little advice. Sleep is a much more powerful tool than you may imagine, so never neglect a good night of sleep in favour of studying (especially the night before the exam). I will also take the liberty of paraphrasing my high school philosophy teacher, Ms. Marques, I hope you will have fun during your exams!

Finally, I would like to thank Paul Tissot-Daguette who did a lot of proofreading of this document. The latter is still not perfect, but it is more readable than it would have been if Paul had not helped me.

Version 2021–12–29

Contents

List of lectures

Chapter 1

Summary by lecture

Lecture 1 : Introduction to propositional logic — Tuesday 21st September 2021 ________ *p. 15*

- Explanation of the course organisation.
- Small summary of the history around propositional logic.
- Definition of the concept of propositions.
- Definition of connectives (negation, conjunction, disjunction, exclusive or, implication, biconditional) and their precedence.
- Explanation of how truth tables work.

Lecture 2 : Logical equivalences and normal forms — Wednesday 22nd September 2021 __ *p. 21*

- Definition of satisfiability (tautology, contingency, contradiction, satisfiable, unsatisfiable).
- Definition of logical equivalences.
- Explanation of the most important logical equivalences and of equivalence proofs.
- Definition of normal forms and explanation of the DNF (Disjunctive Normal Form) and the CNF (Conjuctive Normal Form).

Lecture 3 : Introduction to predicate logic — Tuesday 28th September 2021 ________ *p. 27*

- Introduction and history of predicate logic.
- Definition of the concept of variables, predicates and propositional functions.
- Definition of quantifiers (universal, existential, and uniqueness).
- Definition of the validity of a statement (valid, satisfiable, or unsatisfiable).

Lecture 4 : Nested quantifiers, and negating them — Wednesday 29th September 2021 __ *p. 32*

- Explanation on how to negate quantifiers, using the De Morgan's Laws for quantifiers.
- Explanation on how to have nested quantifiers.
- Explanation of the argument form.
- Summary of the different methods we have to build knowledge.

Lecture 5 : Inference rules and the beginning of proofs — Tuesday 5th October 2021 __ *p. 37*

- Explanation of the different inference rules in propositional logic.
- Explanation of the different inference rules in predicate logic.
- Definition of the vocabulary used for proofs (theorem, axiom, lemma, corollary, proposition, conjecture).

- Definition of the different types of proofs: direct and indirect (contrapositive or contradiction).

Lecture 6 : The end of proofs and the beginning of sets — Wednesday 6th October 2021 . *p. 43*

- Examples of different proofs (direct, by contraposition and by contradiction).
- Explanation of other proof methods, for biconditional statements, proofs by case, proof by counterexample and existence proofs (constructive or non-constructive).
- Examples of some mistakes (or voluntary “mistakes”) that can append in proofs.
- Definition of the concept of sets, and of the most important ones (sets of numbers, intervals, universal and empty sets).

Lecture 7 : The end of sets and the beginning of functions — Tuesday 12th October 2021 *p. 51*

- Definition of set equality and subsets.
- Definition of power sets, tuples, Cartesian product, truth sets of predicates and set cardinality.
- Definition of the set operations, i.e. union (and its cardinality), intersection, difference, complement and symmetric difference.
- Explanation on how to prove set equalities.
- Definition of functions and explanation of some of its terminology.
- Definition of injective (one-to-one), surjective (onto) and bijective functions.

Lecture 8 : End of functions and beginning of relations — Wednesday 13th October 2021 *p. 58*

- Definition of inverse, composed and partial functions.
- Definition of relations, and their compositions.
- Explanation of relations on a set, and their possible properties (reflexive, symmetric, antisymmetric and transitive).

Lecture 9 : Equivalence relations and partial orderings — Tuesday 19th October 2021 *p. 64*

- Definition of equivalence relations
- Definition of equivalence classes, and explanation of their equivalence to set partitions.
- Definition of partial orderings (poset), and Hasse Diagrams.
- Definition of total ordered and well-ordered sets.

Lecture 10 : Sequences and countable infinities — Wednesday 20th October 2021 *p. 68*

- Definition of least upper, greatest lower bound and lattices.
- Definition of lexicographic ordering.
- Definition of sequences, and explanation of the most important ones (arithmetic progression, geometric progression and recurrence relations).
- Explanation on how to compute sums and products.
- Definition of set cardinality, and of countable sets.
- Proof that $\mathbb{N}$ and $\mathbb{Q}$ are countable, but $\mathbb{R}$ is not.

Lecture 11 : Introduction to algorithms — Tuesday 26th October 2021 *p. 73*

- History and definition of algorithms.
- Explanation of two searching algorithms: linear and binary search.

Lecture 18 : Basis, arithmetic, and primes — Wednesday 17th November 2021 *p. 111*

- Explanation of the concept of basis, and how to convert from one basis to another.
- Explanation on how to do arithmetic (addition and multiplication) with base b expansions.
- Explanation on how to compute $b^n \bmod n$.
- Definition of prime and composite numbers. Explanation and proof of the Fundamental Theorem of Arithmetic, the Trial Division theorem, and the infinitude of primes.
- Explanation of the sieve of Eratosthenes.

Lecture 19 : Euclid is the best, and we learn to count — Tuesday 23rd November 2021 *p. 117*

- Definition of GCD and LCM.
- Explanation of the Euclidean algorithm, and of the link between the GCD and the LCM of two numbers.
- Explanation of some facts about number theory, which will not be at the exam.
- Explanation of the product, addition and soustraction rules.
- Definition of permutations and combinations.

Lecture 20 : Repetitions and Pascal's triangle — Wednesday 24th November 2021 *p. 123*

- Proof of the theorem for the computation of r-combinations.
- Definition of r-permutations with repetitions, and proof of the theorem for computing them.
- Definition of r-combinations with repetitions, and proof of the theorem for computing them.
- Definition of permutations with repeated items, and proof of the theorem for computing them.
- Explanation of the binomial theorem and of Pascal's identity.

Lecture 21 : Pigeons, pigeons, holes and recursive counting — Tuesday 30th November 2021 *p. 130*

- Explanation of the pigeon-hole principle, its generalised version, and of examples in which it applies.
- Explanation on how to count with recurrence relations.

Lecture 22 : Closed forms of recurrence relations — Wednesday 1st December 2021 *p. 134*

- Explanation on how to solve linear homogeneous recurrence relation of degree k with constant coefficients, and definition of its characteristic equation.
- Definition of generating functions.
- Explanation on how to solve recurrence relations using generating functions.

Lecture 23 : Alienating sets, and *désolé pour le dérangement* — Tuesday 7th December 2021 *p. 139*

- Explanation on how to use generating functions to compute combinations, combinations with repetitions, and some more generalised cases.
- Definition of the extended binomial coefficients, and explanation of the extended binomial theorem.
- Explanation of the principle of Inclusion-Exclusion.
- Definition of derangements, and explanation on how to compute them.

Tuesday 21st September 2021 — **Lecture 1 : Introduction to propositional logic**

Chapter 2

Introduction and organisation

2.1 Content

General information The course's name was changed recently from Discrete Mathematics; so this will basically be Discrete Mathematics. The teacher's name is Karl Aberer, and it's email is karl.aberer@epfl.ch.

Subject of the course We mostly speak about **discrete mathematics**, structures of which the number of elements is finite or can be enumerated. For example, natural numbers, graphs, formulas (since they have a finite number of symbols).
However, we will not study real numbers. Those will be the subject of the Analysis and Linear Algebra Courses.

Why important Computers are discrete objects: it's components have discrete functions, all data are discrete objects, programs are discrete objects. Everything that is touched by a computer is discrete.

Content The course follows the book "Discrete Mathematics and Its Applications" by Kenneth H. Rosen.

Relation to other courses We will see some core concepts of mathematics (such as sets, functions, relations, . . .). We see them in other course, but we need to understand them for discrete mathematics, they are fundamental so it is good to see them more than once, and it teaches us English.

2.2 Organisation (Covid)

Hybrid course It is a hybrid course, it is subject to evolution. For exercices, there are 7 exercise rooms, and online via Discord.

Online platforms

- **Moodle** Official announcements
- **Zoom** Used for streaming the lectures, Q&A, recoding, polls
- **Switchtube** Recorded lectures
- **Discord** For exercises and interacting with assistants
- **Piazza** For discussing and resolving questions, among students and with assistants. It is organised by weeks (see folders on the top).
- **Kahoot** For weekly (non-graded) quizzes on Tuedays.

Exercices They are distributed at the beginning of the week. Open questions aren't questions that could come up in exams, whereas exam questions are example of exams. It is normal if we cannot answer every question, don't panic, it's not a problem.
Rooms will be assigned soon on Moodle.

Attendance If we don't want to come to the lectures or the exercises, we are allowed not to come. It's however highly recommended. Anyhow, we can use the following platforms to ask questions:

- **Moodle** Organisation
- **Zoom** During lessons
- **Discord** Interact with assistants online and offline
- **Piazza** Interact with other students, and the teaching team

Advice

- Do not procrastinate. Time flies and does not come back.
- Ask questions.
- Do exercises.

Exam There will be one exam, and a midterm exam (the latter may or may not be graded). They will be MCQ, with only one right answer. If we want, we can give more than one answer, but it will diminish the number of points.
It will be on the 28th of January, from 8:15 to 11:15.

Chapter 3

Propositional logic

3.1 Introduction

Introduction

Logic is the language of mathematics, we need it to make human language precise. There are problems in natural language when interpreting expressions such as “or” or “if . . . then”.

Logic is the basis for mathematical proofs, automated reasoning (neural networks), and so on. It is omnipresent in computing.

Logic is about statements that are either **true or false**. **Propositional logic** is the most basic form of logic.

Some history

Greek philosophers created a lot of formalism that looks the same as what we have today, but they still struggled on some concepts. Aristotle and Chrysippus are example of such philosophers.

Modern mathematicians, such as Leibniz (1750), Boole (1860) and De Morgan (1870), formulated propositional logic. Even though this kind of logic is very basic, it introduces many fundamental concepts for mathematics, such as formal language, variables and operators, axioms, inference, proof, and truth values.

It took humans 3000 years to formalise, so even if it will look simple it is not

Propositional logic and computing

Propositional logic allows us to formulate basic search queries (for Google, for instance), describe computer circuits, specify properties of software systems, formally describe games such as Sudoku, and so on.

There exists an algorithm that can decide whether something that can be stated in propositional logic is true or false, automatically. This is not the case for other logics.

Propositions

We define a **proposition** to be a declarative sentence that is either **true or false** (this is very important). For example, “the moon is made of green cheese”, “the Earth is round”, “$1 + 0 = 1$” and “$0 + 0 = 2$” are propositions.

However, orders or questions are not propositions. For example, the following sentences are not propositions: “Sit down!”, “What time is it?”, “$x + 1 = 2$”, “$x + y = z$”.

Propositional logic uses, of course, propositions.

Atomic propositions

Atomic propositions are propositions that cannot be expressed in terms of simpler propositions. For example, “the moon is made of green cheese” is an atomic proposition, we cannot “look inside”.

We use letters such as $p, q, r, s, \ldots$ to express **propositional variables**. We could for example use p to denote $p =$ “The Earth is round”.

The proposition that is always true is denoted by **T**, and the one that is always false is denoted by **F**.

3.2 Logical connectives

Compound propositions

Compound propositions are constructed from **logical connectives** and other propositions. For example:

$$p \leftrightarrow q \land \neg r$$

The most important logical connectives (which we will explain after) are:

- Negation
- Conjunction
- Disjunction
- Exclusive or
- Implication
- Biconditional

Truth table

A **truth table** lists all possible truth values of the propositional variable occurring in a compound propositions, and corresponding truth values of the compound propositions.
We will see examples hereinafter.

Negation

Let p be a proposition. The **negation** of p, denoted $\neg p$ (or $\overline{p}$) is the statement "it is not the case that p". We read $\neg p$ "not p".
For example, if $p =$ "the Earth is round" (which is true (or is it ????? (okay that's a joke, I mean I'm not at EPFL thinking the Earth is flat))), then $\neg p =$ "the Earth is not round" (which is false).
Here is its truth table:

p	$\neg p$
T	F
F	T

In other words, something that is true becomes false, and something false becomes true.

Conjunction

Let p and q be propositions. The **conjunction** of p and q, denoted $p \land q$, is the proposition "p and q". The conjunction $p \land q$ is true when both p and q are true and is false otherwise.
For example, if we have $p =$ "the Earth is round" (true), $q =$ "the moon is round" (true), and $r =$ "the moon is made of green cheese" (false), then $p \land q$ is true and $p \land r$ is false.
Here is its truth table:

p	q	$p \land q$
T	T	T
F	T	F
T	F	F
F	F	F

Disjunction

Let p and q be propositions. The **disjunction** of p and q, denoted $p \lor q$, is the proposition "p or q". The disjunction $p \lor q$ is false whenever p and q are false, and is true otherwise. In other words, it is true whenever either p or q or both is true.
Using the same examples as above (p and q true, r false), we have that both $p \lor q$ and $p \lor r$ are true.
Here is its truth table:

p	q	$p \lor q$
T	T	T
F	T	T
T	F	T
F	F	F

Exclusive or

Let p and q be propositions. The **exclusive or**, also called xor, of p and q, denoted $p \oplus q$, is the proposition "p or q, but not both". The exclusive or $p \oplus q$ is true when one of p and q is true, but not both.
Here is its truth table:

p	q	$p \oplus q$
T	T	F
F	T	T
T	F	T
F	F	F

Or in natural language

In natural language, "or" has two meanings. The inclusive or (disjunction) and the exclusive or (xor) can be both represented using the word "or".
For example, if we say "candidates for this position should have a degree in mathematics or computer science", it means that candidates must have one of those degrees, but they may also have both. On the other hand, if at a restaurant we are asked whether we want "a soup or a salad" with our entrée, we are not expected to answer "both".

Implications

Let p and q be propositions. The **conditional statement** $p \to q$ is the proposition "if p, then q". The conditional statement $p \to q$ is false when p is true and q is false, and is true otherwise. We call p the **premise** (or hypothesis, or antecedent), and we call q the **conclusion** (or consequence).
For example, if we have $p =$ "the Earth is round" (true), $q =$ "the moon is round" (true) and "the moon is made of green cheese" (F), then $p \to q$ is true, $p \to r$ is false and $r \to p$ is true. When we suppose something false, then anything can be true.
Here is its truth table:

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

Intuition

It is important to understand that implication does not require any connection between the antecedent and the consequence. For example, the following implication is true, even though it would never come up in a conversation (except between AICC student): "if the moon is made of green cheese, then I have more money than Bill Gates".

When doing maths, we need to prove the implication, and thus we need an implication between the propositions (it's a bit hard to prove an implication between two completely unrelated things).

We can also understand implication as some kind of obligation or contract. For example, if a politician says "if I am elected, then I will lower taxes". If they are elected but do not lower taxes, people will say that they broke their campaign pledge. If they are not elected, then no one cares.

We can notice that if the premise is false, then the conditional statement is true. Similarly, if the conclusion is true, then the conditional statement is true.

Fallacy

Reversing the direction of the arrow is a common logical fallacy, $p \to q$ is not the same as $q \to p$.

For example, saying that "if it is a dog, then it has four legs" is not the same as saying "if it has four legs, then it is a dog".

Mathematical language

In natural language, there are plenty ways of saying implications. The most important ones are "p is sufficient for q" and "q is necessary for p". Those are the sentences we use in maths.

We say that **a necessary condition** for p is q, and **a sufficient condition** for q is p.

For example, we know that "If Shaun is a sheep, then it is a mammal". So, a necessary condition for Shaun to be a sheep, is that it is a mammal (if it were not a mammal, then we would know that it is not a sheep). Similarly, a sufficient condition for Shaw to be a mammal is to be a sheep (we know that if it is a sheep, then it is definitely a mammal, but if Shaun is not a sheep, then it may also be a mammal).

Another example would be the following: If my age is more that 20 (p), then ($\rightarrow$) my age is more than 10 (q). So, a sufficient condition for my age being more than 10, is that I am more than 20 and a necessary condition for my age being more that 20, is that I am more than 10.

Similarly, let's say that if it rains, then I take an umbrella. So, a sufficient condition to take an umbrella is that it rains, or, in other words, a necessary condition that it rains, is that I take an umbrella.

Biconditional

Let p and q be propositions. The **biconditional statement** $p \leftrightarrow q$ is the propositions "p if and only q" (p iff q). The biconditional statement $p \leftrightarrow q$ is true whenever p and q have the same truth values, and is false otherwise. Biconditional statements are also called bi-implications (since it is $p \rightarrow q$ and $q \rightarrow p$). We can also say p is necessary and sufficient for q.
Here is its truth table:

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Natural language

Sometimes, in natural language, we use implications when we really mean biconditionals. For example, when we say "if you finish your meal, then you can have a dessert", we also mean "if you do not finish your meal, the you will not have a dessert", so it is a biconditional.

Precedence

To avoid having to use thousands of parenthesis, we use the following rule of precedence (priority of operations):

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

We can thus write:

$$(\neg p) \vee q \equiv \neg p \vee q$$

$$p \rightarrow (q \vee (\neg r)) \equiv p \rightarrow q \vee \neg r$$

Notice that the exclusive or does not have any precedence rule. Thus, we must always use parenthesis for this operation.

Wednesday 22^{nd} September 2021 — **Lecture 2 : Logical equivalences and normal forms**

Truth tables for compound proposition To construct a truth table, we make a row for every possible combination of values for the atomic propositions. We then make a column for the truth values of each sub-expression that occurs in the compound proposition. Finally, we make a column for the compound proposition.
For example, we can find the truth table of $(p \lor \neg q) \to (p \land q)$:

p	q	$\neg q$	$p \lor \neg q$	$p \land q$	$(p \lor \neg q) \to (p \land q)$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	F	F	F	T
F	F	T	T	F	F

Satisfiability Depending on the values the compound expression can take, we define the concept of **tautology**, **contingency**, **contradiction**, **satisfiable** and **unsatisfiable**:

Compound expression	**Called**	**Example**	**Satisfiability**
All true	Tautology	$p \lor \neg p$	Satisfiable
True or false, depending on the row	Contingency	p	
All false	Contradiction	$p \land \neg p$	Unsatisfiable

For the contingencies, we also, rarely, say that the expression is “contingent on the values”.

Example Let’s say we want to determine the satisfiability of the following compound proposition:

$$(p \lor \neg q) \land (q \lor \neg r) \land (r \lor \neg p)$$

We see that with $p \equiv T$, $q \equiv T$ and $r \equiv T$, this proposition is true. Thus, it is satisfiable.
Let’s now determine whether the following expression is satisfiable:

$$(p \lor q \lor r) \land (\neg p \lor \neg q \lor \neg r)$$

We can take $p \equiv T$ and $q \equiv F$ (r does not matter) to make this expression true. This implies that this expression is satisfiable.
We see that we do not need to make a truth table to prove that an expression is satisfiable (we may need one to prove that it is unsatisfiable, however).

Boolean queries An example of utility for proposition logic can be useful is boolean queries, for Google for instance. When we write “Lausanne university”, Google will translate it to “the document contain ‘Lausanne AND university’”. Similarly, if we write “Lausanne university|studying -UNIL”, Google will translate it to “Lausanne AND (university OR studying) and NOT UNIL”.

3.3 Logical equivalences

Definition of logical equivalence Two compound propositions p and q are **logically equivalent** if $p \leftrightarrow q$ is a tautology. We write $p \equiv q$.

> *Intuition* Two compound propositions p and q are equivalent if and only if the columns in a truth table are the same.

Example We can show that $\neg p \lor q \equiv p \to q$ using a truth table:

p	q	$\neg p$	$\neg p \lor q$	$p \to q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

This means that we actually do not need the implication symbol.

De Morgan's Laws

De Morgan's Laws state that:

$$\neg(p \land q) \equiv \neg p \lor \neg q$$

$$\neg(p \lor q) \equiv \neg p \land \neg q$$

Proof of the first law

we can make a truth table:

p	q	$\neg p$	$\neg q$	$p \land q$	$\neg p \lor \neg q$	$\neg(p \land q)$
T	T	F	F	T	F	F
T	F	F	T	F	T	T
F	T	T	F	F	T	T
F	F	T	T	F	T	T

Equivalences with basic connectives

The following equivalences using only $\land$ and $\lor$ are really important and useful:

Equivalence	**Name**
$p \land T \equiv p$ $p \lor F \equiv p$	Identity laws
$p \lor T \equiv T$ $p \land F \equiv F$	Domination laws
$p \lor p \equiv p$ $p \land p \equiv p$	Idempotent laws
$\neg(\neg p)$	Double negation law
$p \lor (p \land q) \equiv p$ $p \land (p \lor q) \equiv p$	Absorption laws
$p \lor \neg p \equiv T$ $p \land \neg p \equiv F$	Negation laws
$p \lor q \equiv q \lor p$ $p \land q \equiv q \land p$	Commutative laws
$(p \lor q) \lor r \equiv p \lor (q \lor r)$ $(p \land q) \land r \equiv p \land (q \land r)$	Associative laws
$p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$ $p \land (q \lor r) \equiv (p \land q) \lor (p \land r)$	Distributive laws

Equivalences with implications

The following equivalences using implications are less important, but still it is interesting to see them:

$p \to q \equiv \neg p \lor q$ $p \to q \equiv \neg q \to \neg p$
$p \lor q \equiv \neg p \to q$ $p \land q \equiv \neg(p \to \neg q)$ $\neg(p \to q) \equiv p \land \neg q$
$(p \to q) \land (p \to r) \equiv p \to (q \land r)$ $(p \to r) \land (q \to r) \equiv (p \lor q) \to r$ $(p \to q) \lor (p \to r) \equiv p \to (q \lor r)$ $(p \to r) \lor (q \to r) \equiv (p \land q) \to r$

$p \leftrightarrow q \equiv (p \to q) \land (q \to p)$ $p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$ $p \leftrightarrow q \equiv (p \land q) \lor (\neg p \land \neg q)$ $\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

They can be found easily using the equivalences from the last paragraph, and from:

$$p \leftrightarrow q \equiv (p \to q) \land (q \to p)$$

$$p \to q \equiv \neg p \land q$$

Contrapositive, converse and inverse

We say that $\neg q \rightarrow \neg p$ is the **contrapositive** of $p \rightarrow q$. We have the following equivalence:

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$

We also define that $q \rightarrow p$ is the **converse** of $p \rightarrow q$, and $\neg q \rightarrow \neg p$ is the **inverse** of $p \rightarrow q$ (and the contrapositive of $q \rightarrow p$; in other words, the contrapositive of the converse is called the inverse). The converse and the inverse are equivalent, but they are not equivalent to $p \rightarrow q$ (this is a common fallacy politicians use).

Applying logical equivalences

The proposition in a known equivalence can be replaced by any compound proposition.
For example, we know that:

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$

Then, picking $p \equiv (p_1 \lor p_2)$ and $q \equiv (q_1 \land q_2)$:

$$(p_1 \lor p_2) \rightarrow (q_1 \land q_2) \equiv \neg(q_1 \land q_2) \rightarrow \neg(p_1 \lor p_2)$$

Constructing new logical equivalences

We know that the logical equivalence is transitive. In other word, we can show that two expressions are logically equivalent by developing a series of logically equivalent statements; to prove $A \equiv B$ we make such a series, beginning with A and ending with B:

$$A \equiv A_1, A_1 \equiv A_2, \ldots, A_n = B \implies A \equiv B$$

This is called an **equivalence proof**, and it is much more efficient than using truth tables. However, it cannot be done automatically, we need inspiration.

Example 1

Let's say we want to show the following equivalence:

$$\neg(p \lor (\neg p \land q)) \equiv \neg p \land \neg q$$

We can use the following steps:

$$\begin{aligned}
\neg(p \lor (\neg p \land q)) &\overset{\text{De Morgan}}{\equiv} \neg p \land \neg(\neg p \land q) \\
&\overset{\text{De Morgan}}{\equiv} \neg p \land (p \lor \neg q) \\
&\overset{\text{Distributivity}}{\equiv} (\neg p \land p) \lor (\neg p \land \neg q) \\
&\equiv F \lor (\neg p \land \neg q) \\
&\equiv \neg p \land \neg q
\end{aligned}$$

Example 2

Let's say we want to show that $(p \land q) \rightarrow (p \lor q)$ is a tautology:

$$\begin{aligned}
(p \land q) \rightarrow (p \lor q) &\equiv \neg(p \land q) \lor (p \lor q) \\
&\equiv (\neg p \lor \neg q) \lor (p \lor q) \\
&\equiv (\neg p \lor p) \lor (\neg q \lor q) \\
&\equiv T \lor T \\
&\equiv T
\end{aligned}$$

Example 3

We want to show that $(p \rightarrow r) \land (q \rightarrow r) \equiv (p \lor q) \rightarrow r$:

$$\begin{aligned}
(p \rightarrow r) \land (q \rightarrow r) &\equiv (\neg p \lor r) \land (\neg q \lor r) \\
&\equiv (\neg p \land \neg q) \lor r \\
&\equiv \neg(p \lor q) \lor r \\
&\equiv (p \lor q) \rightarrow r
\end{aligned}$$

Example 4

We wonder if $\oplus$ is distributive (with xor, we always use parenthesis, since there exists no clear precedence rule):

1. $p \land (q \oplus r) \equiv (p \land q) \oplus (p \land r)$

2. $p \vee (q \oplus r) \equiv (p \vee q) \oplus (p \vee q)$

We can prove that the first one holds, but the second one is wrong: we can let $p \equiv T, q \equiv T, r \equiv T$. On the left hand side, we get:

$$T \vee (T \oplus T) \equiv T \vee F \equiv T$$

On the right hand side, we get:

$$(T \vee T) \oplus (T \vee T) \equiv T \oplus T \equiv F$$

3.4 Normal forms

Introduction

We use normal forms to convert arbitrary propositional statements into canonical form; if two propositions are equivalent then they have the same normal form. This is useful for the automated proofing of theorems (we can put both expressions in their normal forms and see if they are equal). This is also the basis for the formulation of some of the most central problems of complexity theory, and it finds some applications for circuit design. Those applications will be further discussed in other courses.

Definition of the Disjunctive Normal Form

A propositional formula is in **Disjunctive Normal Form (DNF)** if it only consists of a disjunction of compound expressions, where each compound expression consists of a conjunction of a set of propositional variables or their negation.

Example

For example, the following expressions are in DNF:

$$(p \wedge \neg q) \vee (\neg p \wedge q), \quad p \vee (\neg p \wedge q), \quad (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge r)$$

The following expressions are *not* in DNF:

$$(p \wedge \neg q) \vee \neg(\neg p \wedge q), \quad (p \vee q) \wedge (\neg p \wedge q), \quad \neg(\neg p \vee q)$$

Construction of DNF

Every compound proposition can be put in DNF using the following procedure:

1. Construct the truth table for the proposition.
2. Select the rows that evaluate to T.
3. For each of the propositional variables in the selected rows, add a conjunction which includes the positive form of the propositional if the variable is assigned T in that row, or the negated form if the variable is assigned F in that row.

We can then simplify further the resulting proposition using the following equivalence:

$$(p \wedge q) \vee (p \wedge \neg q) \equiv p$$

Example

Let's say we want to find the DNF of the following expression

$$(p \vee q) \to \neg r$$

First, we make its truth table:

p	q	r	$\neg r$	$p \vee q$	$(p \vee q) \to \neg r$
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	T	T	F
F	T	F	F	T	T
F	F	T	T	F	T
F	F	F	F	F	T

So, its DNF is given by:

$$(p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$$

We can turn it back to its “simplified form”. Let’s first see the following simplification:

$$\underbrace{(p \land q \land \lnot r) \lor (p \land \lnot q \land \lnot r)}_{\equiv p \land \lnot r} \lor (\lnot p \land q \land \lnot r) \lor (\lnot p \land \lnot q \land r) \lor (\lnot p \land \lnot q \land \lnot r)$$

We can do another simplification than the one written as an underbrace. We know that $p \lor p \equiv p$, so if we look at the three last terms, we know they are equivalent to:

$$\overbrace{\underbrace{(\lnot p \land q \land \lnot r) \lor (\lnot p \land \lnot q \land \lnot r)}_{\equiv \lnot p \land \lnot r} \lor \underbrace{(\lnot p \land \lnot q \land r) \lor (\lnot p \land \lnot q \land \lnot r)}_{\equiv \lnot p \land \lnot q}}^{\equiv (\lnot p \land q \land \lnot r) \lor (\lnot p \land \lnot q \land r) \lor (\lnot p \land \lnot q \land \lnot r)}$$

We can then simplify our original proposition to:

$$(p \land \lnot r) \lor (\lnot p \land \lnot r) \lor (\lnot p \lor \lnot q) \equiv \lnot r \lor (\lnot p \land \lnot q) \equiv (p \lor q) \to \lnot r$$

Definition of the Conjunctive Normal Form

We say that a compound proposition is in **Conjunctive Normal Form (CNF)** if it consists of a conjunction of compound expressions, where each compound expression consists of a disjunction of a set of propositional variables or their negations.

Example

The following expressions are in CNF:

$$(p \lor q) \land (\lnot p \lor q), \quad (p \lor q \lor \lnot r) \land (\lnot p \lor \lnot q \lor \lnot r)$$

The following expressions are *not* in CNF:

$$(p \land \lnot q) \lor (\lnot p \land q), \quad p \lor (\lnot p \land q), \quad (p \land \lnot q) \lor \lnot(\lnot p \land q), \quad \lnot(\lnot p \lor q)$$

Construction of the CNF

One of the ways to construct a CNF, is to:

1. Eliminate implications.
2. Move negations inward.
3. Use distributive and associative laws.

Another way is to see that $p \equiv \lnot(\lnot p)$ and that the negation of a DNF is a CNF (but of the negated expression, they are not equivalent) by De Morgan’s Laws. Thus we can find the DNF of $\lnot p$, meaning doing the exact same thing on the truth table, but on the rows where p is false, and then take the negation on it.

Example

Let’s say that we want to find the CNF of the following expression:

$$\lnot(p \to q) \lor (r \to p)$$

For the first method, we can use the fact that $(a \land b) \lor c \equiv (a \lor c) \land (b \lor c)$:

$$\begin{aligned} \lnot(p \to q) \lor (r \to p) &\equiv \lnot(\lnot p \lor q) \lor (\lnot r \lor p) \\ &\equiv (p \land \lnot q) \lor (\lnot r \lor p) \\ &\equiv (p \lor \lnot r \lor p) \land (\lnot q \lor \lnot r \lor p) \\ &\equiv (\lnot r \lor p) \land (\lnot q \lor \lnot r \lor p) \\ &\equiv (p \lor \lnot r) \land (p \lor \lnot q \lor \lnot r) \end{aligned}$$

For the second method, we need to draw its truth table:

p	q	r	$\lnot(p \to q) \lor (r \to p)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	T

So, picking only the raws where the expression is negative, we get:

$$\lnot((\lnot p \land q \land r) \lor (\lnot p \land \lnot q \land r)) \equiv \lnot(\lnot p \land r) \equiv (p \lor \lnot r)$$

Complexity of DNF and CNF

We realise that both DNF and CNF can be much larger than the original proposition. More precisely, there exist cases of propositions with n clauses for which the DNF has 2^n clauses, and similarly for CNF.
We can create very complicated scenarios by concatenating $\wedge(p_{n+1} \vee q_{n+1})$ at the end of a DNF: if we want decide to turn it to a CNF, this will be very complicated: we will end up with twice the number of starting elements. We can do it again to reach more and more complicated expressions, having always twice as many clauses as the one before.

Example of application of propositional logic

Let's say that on an island there are knights who always tell the truth and knaves who always lie. We meet A and B. A says that B is a knight, and B says that both of them are of opposite type.
Let's pick $p :=$ A is a knight (tells the truth) and $q :=$ B is a knight (lies).
Assume A is a knight, so p is true. Then B is a knight, according to what A says, so q is T. What B says is that $p \oplus q$ is T. This is a contradiction.
Let's now assume that A is a knave, so p is F. Thus, q is F, and what B says works.

Tuesday 28th September 2021 — **Lecture 3 : Introduction to predicate logic**

Chapter 4

Predicate logic

4.1 Introduction

Introduction

We realise that propositional logic is not enough.
For example, if we know that “all men are mortal” and “Socrates is a man”, we cannot say that “Socrates is mortal” using propositional logic. Indeed, we would have to “look inside” the propositions, but we cannot since sentences are either always true or always false.
We will need to go further than propositional logic, with predicate logic. This kind of logic introduces a concept of quantifiers, and they allow us to overcome the limitations of the Aristotelian logic.

Small history

Aristotle developed a (limited form of) predicate logic. It was then independently developed by Frege and Pierce between the 19th and the 20th century.
Thus, it took also around 2000 years to develop.

Impact in mathematics

Predicate logic (also called first-order logic) is the standard language to represent mathematical statements.
Some mathematicians such as Hilbert hoped that this would be a complete system of logic to describe mathematics. However, in the twentieth century, Gödel proved that it is not possible not prove everything using predicate logic: this is Gödel’s incompleteness theorem.

Impact in computer science

Predicate logic is very important in computer science. Indeed, we find it when formulation general search queries in databases, in logic programming (such as Prolog), automated theorem proving, software verification, symbolic AI systems, and so on.

Variables

As explained above, we want to characterise an object by its properties. Let’s call that object x; this is a **variable**.
We could then write $\text{man}(x)$ to say “x is a man”, or $x > 3$ to say “x is a number larger than 3”.

Definitions

We define **predicates** to be statements that contains a variable.
For example, the following statements are predicates:

$$x > 3, \quad x = y + 3, \quad x + y = z$$

Its variables can be replaced by a value from a domain U, the **universe of discourse**, for example the integers. Depending on the value that the variable takes, the predicate becomes a proposition (from propositional logic) which is either true or false. Thus, connectives from propositional logic can be applied to predicate logic.

Example 1

Let $R(x, y, z) := x + y = z$ and the domain be integers.
$R(2, -1, 5)$ is false since $2 + (-1) \neq 5$; $R(3, 4, 7)$ is true since $3 + 4 = 7$; and we don't know for $R(x, 3, z)$ since $x + 3 = z$ can be true or false.

Example 2

Let $P(x) := x > 0$. Then, the following propositions are true:

$$P(3) \vee P(-1), \quad P(3) \to \neg P(-1)$$

And, the following propositions are false:

$$P(3) \wedge P(-1), \quad P(3) \to P(-1)$$

Propositional functions

We call **propositional functions** expressions constructed from predicates and logical connectives containing variables.
For example, the following expressions are propositional functions:

$$R(x, y) := P(x) \to P(y), \quad R(y) = P(3) \wedge P(y)$$

4.2 Quantifiers

Quantifiers

We need **quantifiers** to express to which extent a propositional functions is true. Indeed, depending on what the value of the variables are, the propositional function can have different values. This is captured by the quantifiers.
For example, $x > 0$ is true for 1, 2 and so on, but not for 0, -1, -2, ... However, $x < x - 1$ is never true and $x < x + 1$ is always true.
Such quantifiers can be used as logical connective.

Universal quantifier

The universal quantification of a propositional function $P(x)$ is the statement "$P(x)$ is true for all values x from its domain U". We write this $\forall x\ P(x)$, and read "for all x, $P(x)$". We call $\forall$ the **universal quantifier**.
In other words, $\forall x\ P(x)$ means that for any x in the universe, the proposition evaluates to true.

Example

If $P(x) := x > 0$ and U is the integers, then $\forall x\ P(x)$ is false, because $P(0)$ is false. Now, if we take the same predicate but with U being the positive integers (note that ***0 is neither positive nor negative*** in this course), then $\forall x\ P(x)$ is true.
Thus, it is important to notice that the result of a statement can depend on the universe we choose.

Existential quantifier

The existential quantification of a propositional function $P(x)$ is the statement "there exists an element x from the domain U such that $P(x)$ is true". We write $\exists x\ P(x)$, and it reads "for some x, $P(x)$", or "for at least one x, $P(x)$". We call $\exists$ the **existential quantifier**.

> *Link with universal quantifier*
>
> We see that if $\forall x\ P(x)$ is true, then $\exists x\ P(x)$ must also be true (if the domain is not empty). Indeed, if a propositional function is true for every element, then there definitely exists an element for which the propositional function is true.

Empty domains

Note that when the domain is empty, weirder things appear. Indeed, $\forall x\ P(x)$ is true (every elements from the empty domain make $P(x)$ true) but $\exists x\ P(x)$ is false (there is no element from the domain (since it is empty) that makes $P(x)$ true).
Thus, by convention, we never consider empty domains.

Example

This time, if we have $P(x) := x > 0$, then $\exists x\ P(x)$ is true both if U is the integers or if U is the positive integers. However, it is false if U is the negative integers.

Summary

We can draw the following table:

Statement	Is true when	Is false when
$\forall x\ P(x)$	$P(x)$ is true for every x.	There is an x for which $P(x)$ is false.
$\exists x\ P(x)$	There is an x for which $P(x)$ is true.	$P(x)$ is false for every x.

We call a **counterexample** for $\forall x\ P(x)$, a value x for which $P(x)$ is false; and a **witness** for $\exists x\ P(x)$, a value x for which $P(x)$ is true.

Quantifiers with finite domains

We can see that, in fact, if the domain U is finite we do not need such quantifiers. Indeed, for example, if U consists of the integers 1, 2 and 3, then $\forall x\ P(x)$ is equivalent to:

$$P(1) \land P(2) \land P(3)$$

Moreover, $\exists x\ P(x)$ is equivalent to:

$$P(1) \lor P(2) \lor P(3)$$

Uniqueness quantifier

We use $\exists! x\ P(x)$ to express the fact that $P(x)$ is true for one and only one x in the domain U. We usually say "there is a unique x such that $P(x)$", or "there is one and only one x such that $P(x)$".

> *Expression using other quantifiers*
>
> This is skipping a bit ahead, but we can express the uniqueness quantifier using other quantifiers:
>
> $$\exists! x\ P(x) \equiv \exists x (P(x) \land \forall y (y \neq x \to \neg P(y)))$$

Example

For instance, if $P(x) := x + 1 = 0$ and U is the integers, then $\exists! x\ P(x)$ is true. However, if $P(x) := x > 0$ with the same domain, then $\exists!\ P(x)$ is false.

Composite statements involving quantifiers

We can apply connectives from propositional logic to predicates. For example:

$$(\forall x\ P(x)) \lor Q(x)$$

Note that this is not a proposition since there is a x in $Q(x)$, which is not bound to a quantifier. Moreover, we have to be careful about the fact that the two x's are different variables: another way of writing this statement would have been $(\forall y\ P(y))Q(x)$.
The quantifiers have higher precedence than all the logical connectives from propositional logic. For example, $\forall x\ P(x) \lor Q(x)$ means $(\forall x\ P(x)) \lor Q(x)$; $\forall x (P(x) \lor Q(x))$ means something different.

Variable binding

We say that a quantifier binds the variable of a propositional function.
$P(x)$ is a propositional function with a **free variable** x, and $\forall x\ P(x)$ is a proposition with **bound variable** x.
Using precedence rule, we see that $\forall x\ P(x) \lor Q(x)$ has a free variable, but $\forall x\ (P(x) \lor Q(x))$ has no free variable.

Validity and satisfiability

A statement involving predicates and quantifiers *with all variables bound* is **valid** if it is true for all domains, and it is true for every propositional function substituted for the predicates in the assertion (in propositional logic we called this a tautology).
Similarly, such a statement is **satisfiable** if it is true for some domains, and if some propositional function can be substituted for the predicates in the assertion.
Otherwise, it is **unsatisfiable**.
In other words, valid means true whatever the variable we choose, satisfiable means that there exists some case in which it is true, unsatisfiable means that it is always false.

Example

For instance, the following statement is valid:

$$\forall x\ \neg S(x) \leftrightarrow \neg \exists x\ S(x)$$

The following statement is satisfiable since we can pick $F(x) \equiv x > 0$ and $T(x) \equiv x > 0$:

$$\forall x(F(x) \leftrightarrow T(x))$$

The following statement is unsatisfiable:

$$\forall x(F(x) \land \neg F(x))$$

Translating from natural language Let's say we want to translate "every student in this class has taken a course in Java" to predicate logic.
Let's first define $J(x) :=$ "x has taken a course in Java" and $S(x) :=$ "x is a student in this class". We must then decide on the domain U. If it is all the students in the class, it is very easy, we can say $\forall x\ J(x)$. If U is all people in the world, then the sentence can be translated to $\forall x(S(x) \rightarrow J(x))$. Note that $\forall x(S(x) \land J(x))$ is incorrect since it would mean that everyone on the Earth is a student and has taken a Java course.

Let's do the same exercise but with "some student in this class has taken a course in Java". Again, if U is all students in the class it is very easy since the sentence translates to $\exists x\ J(x)$. If U is all people in the world, then it translates to $\exists x(S(x) \land J(x))$. Note that $\exists x(S(x) \rightarrow J(x))$ is incorrect since all students could have taken no Java course: there exists somebody on this Earth who is not a student and thus makes $S(x) \rightarrow J(x)$ true.

Similar example Let $P(x)$, $Q(x)$ and $R(x)$ be the statements "x is a clear explanation", "x is satisfactory", and "x is an excuse", respectively. Let's pick all English texts as the domain for x.
"All clear explanations are satisfactory" is translated to:

$$\forall x(P(x) \rightarrow Q(x))$$

"Some excuses are unsatisfactory" is translated to:

$$\exists x(R(x) \land \neg Q(x))$$

"Some excuses are not clear explanations" is translated to:

$$\exists x(R(x) \land \neg P(x))$$

Shorthand notation We can define the following shorthand notations:

$$\forall x \in S\ P(x) \equiv \forall x(x \in S \rightarrow P(x))$$

$$\exists x \in S\ P(x) \equiv \exists x(x \in S \land P(x))$$

This works too for equalities:

$$\forall x \neq 0\ P(x) \equiv \forall x(x \neq 0 \rightarrow P(x))$$

Negations We can notice that this notation is coherent with negations. Indeed, we do have that:

$$\neg\forall x \in S\ P(x) \equiv \neg\forall x(x \in S \rightarrow P(x)) \equiv \neg\forall x(\neg x \in S \lor P(x))$$

Which is equal to:

$$\exists x(x \in S \land \neg P(x)) \equiv \exists x \in S\ \neg P(x)$$

We also find that:

$$\neg\exists x \in S\ P(x) \equiv \neg(\neg\forall x \in S\ \neg P(x)) \equiv \forall x \in S\ \neg P(x)$$

This makes sense, since we have to keep the same hypothesis when negating a statement.

Sudoku Propositional logic and predicate logic are powerful enough to describe games, such as Sudoku.

Let's define $P(i, j, n) :=$ cell(i, j) contains the number n. We then want to encode that "every row contains every number". For example, in the row 1, with the number 1:

$$P(1, 1, 1) \lor P(1, 2, 1) \lor \ldots \lor P(1, 9, 1) = \lor_{j=1}^{9} p(1, j, 1)$$

This is true for every row and for every number, thus it is translated to

$$\land_{i=1}^{9} \land_{n=1}^{9} \lor_{j=1}^{9} P(i, j, n)$$

Using predicate logic, we can simplify the notation:

$$\land_{i=1}^{9} \land_{n=1}^{9} \lor_{j=1}^{9} P(i, j, n) \equiv \forall i \forall n \exists j P(i, j, n)$$

We can do the same reasoning for columns and squares. It would be incredibly hard to translate all our results to a truth table (there would be more rows than atoms in the universe, and if we created one universe for each atoms there would still not be enough atoms), but the point is that it can be translated.

4.3 Logical equivalences in predicate logic

Equivalences Two statements S and T involving predicates and quantifiers are logically equivalent if and only if they have the same truth values no matter which predicates are substituted and which domain of discourse is used.

As mentioned earlier, by convention we do not consider empty domain of discourse.

Example 1 We can draw the following equivalence:

$$\forall x \neg\neg S(x) \equiv \forall x\ S(x)$$

Indeed, $\neg\neg S(x) \equiv S(x)$, and this is independent of the choice of S and of x.

Example 2 Let's say we want to show:

$$\forall x (P(x) \land Q(x)) \equiv \forall x P(x) \land \forall x Q(x)$$

Note that to prove that $a \leftrightarrow b$, it is often easier to show $a \to b$ and $b \to a$.

> *Proof of →* We suppose that $\forall x (P(x) \land Q(x))$ is true.
>
> If a is in the domain, then both $P(a)$ and $Q(a)$ are true. Since they are both true for every element a in the domain, we know that both $\forall x P(x)$ and $\forall x Q(x)$ are true. Thus, $\forall x P(x) \land \forall x Q(x)$ is true.

> *Proof of ←* We suppose that $\forall x P(x) \land \forall x Q(x)$ is true.
>
> If a is in the domain, then both $P(a)$ and $Q(a)$ are true. Thus, $P(a) \land Q(a)$ is true, and we can deduce that $\forall x (P(x) \land Q(x))$ is true.

Distribution of quantifiers of connectives Hereinabove we saw that the universal quantifier is distributive over the and connective. Note that the distribution of quantifiers over connectives is a bit special. Indeed, the following statements are true:

$$\forall x (P(x) \land Q(x)) \equiv \forall x P(x) \land \forall x Q(x)$$

$$\exists x (P(x) \lor Q(x)) \equiv \exists x P(x) \lor \exists x Q(x)$$

However:

$$\exists x (P(x) \land Q(x)) \not\equiv \exists x P(x) \land \exists x Q(x)$$

$$\forall x (P(x) \lor Q(x)) \not\equiv \forall x P(x) \lor \forall x Q(x)$$

Indeed, for both we can use the following counter example: $P(x) :=$ "x is even", $Q(x) :=$ "x is odd" and $\mathbb{Z}$ as the domain.

Wednesday 29th September 2021 — **Lecture 4 : Nested quantifiers, and negating them**

De Morgan's Laws for Quantifiers

We have the following equivalences:

$$\neg\forall P(x) = \exists x \neg P(x)$$

$$\neg\exists P(x) = \forall x \neg P(x)$$

Proof of the first one

We know that $\neg\forall x P(x)$ is true if and only if $\forall x P(x)$ is false. This is equivalent to the existence of an element a such that $P(a)$ is false, and thus $\neg P(a)$ is true. From that, we can deduce that $\exists x \neg P(x)$ is true.

Justification of the name

Those laws are in fact the equivalents of De Morgan's Laws for infinite set. Indeed, if we have $U = \{1, 2, 3\}$, then:

$$\exists x\ P(x) = P(1) \lor P(2) \lor P(3)$$

Thus, negating both side and applying de Morgan's Laws:

$$\neg(\exists x\ P(x)) = \neg P(1) \land \neg P(2) \land \neg P(3) = \forall x \neg P(x)$$

Exercise

Let the following statements:

1. $\exists! x P(x) \to \exists x P(x)$
2. $\forall x P(x) \to \exists! x P(x)$
3. $\exists! x \neg P(x) \to \neg\forall x P(x)$

We wonder what are their truth values.

1. It is valid, since if there exists exactly one, then there exists at least one.
2. It is satisfiable, when there is exactly one element in the domain.
3. Using De Morgan's law, we see that it is equivalent to

$$\exists! x \neg P(x) \to \exists x \neg P(x)$$

   which is equivalent to the first one, so it is valid.

Now, let's consider what happens when we have an empty domain:

1. It is also valid.
2. It is unsatisfiable since we have $T \to F$.
3. It is also valid, since it is equivalent to the first one.

4.4 Nested quantifiers

Nested quantifiers

Nested quantifiers are often necessary to express the meaning of sentences in natural language as well as important concepts in computer science and mathematics.

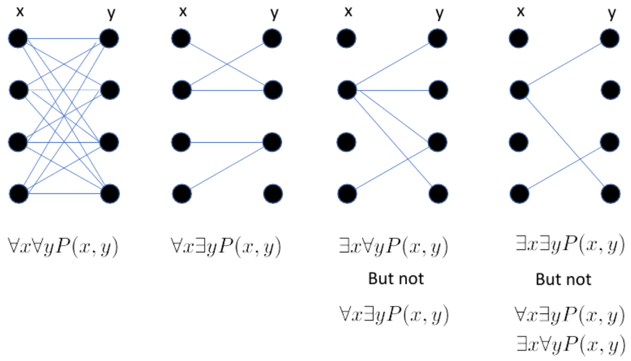
Example

We can translate "every real number has an inverse" as:

$$\forall x \exists y (x + y = 0)$$

where the domains for both x and y are real numbers.

We can draw the following diagram, by linking two points if and only if $P(x, y)$ is true:


x
y
x
y
x
y
x
y
$\forall x \forall y P(x, y)$
$\forall x \exists y P(x, y)$
$\exists x \forall y P(x, y)$
But not
$\forall x \exists y P(x, y)$
$\exists x \exists y P(x, y)$
But not
$\forall x \exists y P(x, y)$
$\exists x \forall y P(x, y)$

Nested propositional functions We can also take **nested propositional functions**.
For example, $\forall x \exists y (x + y = 0)$ can be viewed as $\forall x Q(x)$, where:

$$Q(x) := \exists y P(x, y), \quad P(x, y) := (x + y = 0)$$

Note that $\exists y P(x, y)$ has an unbound variable.

Order of quantifiers The order of quantifiers is really important.
For example, let's take U to be the real numbers, and $P(x, y) := (x + y = 0)$. Then, $\forall x \exists y P(x, y)$ is true (for all number, there does exist an inverse), but $\exists y \forall x P(x, y)$ is false (there does not exist any real number which is an inverse for every number).
We see that it is not always the case. Indeed, if we have $Q(x) := (x + y = y + x)$, then both $\forall x \forall y P(x, y)$ and $\forall y \forall x P(x, y)$ are true (addition is indeed commutative).
In fact, we can always switch the order of the quantifiers whenever we only have universal quantifiers or only existential quantifiers:

$$\forall x \forall y P(x, y) \equiv \forall y \forall x P(x, y)$$

$$\exists x \exists y P(x, y) \equiv \exists y \exists x P(x, y)$$

Translation to natural language Let $F(x, y) :=$ "x and y are friends" and students as the domain. Let's say we want to translate:

$$\exists x \forall y \forall z (F(x, y) \land F(x, z) \land y \neq z \to \neg F(y, z))$$

When translated into natural language, we find: "there exists a student, such that for all of his friends that are different, it is the case that they not friends."
It shows that it is not always evident to convert back and forth, and that our language has a huge flexibility and shortcuts.

Translation from maths Let's say we want to translate "the sum of two positive integers is always positive" into a logical expression.
First, let's rewrite this statement to make the implied quantifiers and domains and domains explicit: "for every two integers, if these integers are both positive, then the sum of these integers is positive".
Second, let's introduce the variables: "for every two integers x and y, if x is positive and y is positive, then $x + y$ is positive".
Third, we can translate it to predicate logic:

$$\forall x \forall y (x > 0 \land y > 0 \to x + y > 0)$$

Translation from natural language Let $L(x, y) :=$ "x loves y".
Let's translate the following sentences:

1. "Everybody loves somebody."
2. "There is someone how is loved by everyone."
3. "There is someone who loves someone."
4. "Everyone loves himself."

We have:

1. $\forall x \exists y L(x, y)$
2. $\exists x \forall y L(y, x)$
3. $\exists x \exists y L(x, y)$
4. $\forall x L(x, x)$

> *Personal remark* I truly hope that $\forall x \exists y (L(x, y) \land L(y, x))$.

Example Let's determine the truth values of the following statements, where the domain is the real numbers:

1. $\forall x \exists y \left(x^2 = y\right)$
2. $\forall x \exists y \left(x = y^2\right)$
3. $\exists x \forall y (xy = 0)$
4. $\exists x \exists y (x + y \neq y + x)$

We have:

1. It is true, we can choose $y = x^2$.
2. It is false, we can pick $x = -1$.
3. It is true, we can choose $x = 0$.
4. It is false, since we know that $\forall x \forall y (x + y = y + x)$ (the addition is commutative), and this is the exact negation.

Prenex Normal Form (not in exams I think) We say that a statement is in Prenex Normal Form (PNF) if and only if it is of the form:

$$Q_1 x_1 Q_2 x_2 \ldots Q_k x_k P(x_1, x_2, \ldots, x_k)$$

where each Q_i is either the existential quantification or the universal quantifier, and $P(x_1, \ldots, x_k)$ is a predicate involving no quantifier.

> *Example* Let's say we have the following proposition:
>
> $$\exists x P(x) \to \exists x Q(x)$$
>
> By doing some tricks (such as introducing another variable for clarity), we see that it is equivalent to:
>
> $$\forall x \neg P(x) \lor \exists y Q(y) \equiv \forall x \exists y (\neg P(x) \lor Q(y))$$
>
> The last one is the PNF.

Chapter 5

Proofs (deriving knowledge)

5.1 Valid arguments

Deriving knowledge Let's say we know something, let's say p (p is true). Then, we may want to extend this knowledge to other facts. We have already seen equivalence proofs, that show that:

$$p \leftrightarrow p'$$

However, we may also want to derive knowledge, which only goes in one way (we are thus going to have something less general). If $p \to q$, then we can deduce that q is true. However, if q is true we cannot deduce that p is true. Those are arguments.

Definitions An **argument** in propositional logic is a sequence of propositions. All but the final proposition are called **premises**, and the last statement is the **conclusion**. The argument is valid if the premises imply the conclusion.
An **argument form** is an argument that is valid no matter what propositions are substituted into its propositional variables. **Inference rules** are simple arguments forms that will be used to construct more complex argument forms.

Modus Ponens We can note:

$$\begin{array}{rl} & p \to q \\ & p \\ \hline \therefore & q \end{array}$$

In other words, if we know that $p \to q$ and p are true, then we can deduce q. Note that $\therefore$ means "therefore" (it must not be confused with $\because$ which means "because"). We call this the **Modus Ponens**, it is a valid argument form.

Example Let's say that we know that "if I have passed AICC, then I can advance to year 2 of the studies" ($p \to q$) and "I have passed AICC" (p). Then we can conclude that "I can advance to year 2 of the studies" (q).
This is a valid argument.

Validity We know that the following proposition is a tautology:

$$(p \land p \to q) \to q$$

So, if we know that p and $p \to q$ are true, then q must necessarily be true (else, the proposition would not be a tautology). Thus, the Modus Ponens is indeed a valid argument form.

Valid argument As mentioned earlier, a valid argument is a sequence of statements, where each statement is either a premise or follows from previous statements by inference rules. It takes the following form:

$$\begin{array}{rl} & p_1 \\ & \vdots \\ & p_n \\ \hline \therefore & q \end{array}$$

We can use the same reasoning as for the Modus Ponens to derive inference rules out of any tautology of the form:

$$(p_1 \land p_2 \land \ldots \land p_n) \to q$$

Example 1 We know the following inference rule (which is Modus Ponens):

$$\begin{array}{rl} & p \\ & p \to q \\ \hline \therefore & q \end{array}$$

So, applying it to some specific propositions, we get the following argument form:

$$\begin{array}{rl} & r \land s \\ & (r \land s) \to q \\ \hline \therefore & q \end{array}$$

Thus, we can derive the following argument:

$$\begin{array}{rl} & p \\ & p \to (r \land s) \\ & (r \land s) \to q \\ & r \land s \\ \hline \therefore & q \end{array}$$

Where the 3 first propositions are premises, and $r \land s$ is found using Modus Ponens.

Usefulness Let's say we know that "If I have passed AICC, Analysis 1, Linear Algebra, etc., then I can advance to year 2 of the studies", and "I have passed AICC", "I have passed Analysis 1", "I have passed Linear Algebra", ...
Then, it seems clear we can deduce that "I can advance to year 2 of the studies". However, we need to show that this argument is valid.

Truth table Let's say we wanted to prove this using a truth table. If we had 20 school subjects, then we would have $2^{20} = 1\,048\,576$ rows, which is not very practical.

Arguments To show this using arguments, we first need to introduce another inference rule, conjunction:

$$\begin{array}{rl} & p \\ & q \\ \hline \therefore & p \land q \end{array}$$

This is based on the following tautology:

$$p \land q \to p \land q$$

We can now use this to find the following argument form:

$$\begin{array}{rl} & (p_1 \land \ldots \land p_n) \to q \\ & p_1 \\ & \vdots \\ & p_n \\ & p_1 \land \ldots \land p_n \\ \hline \therefore & q \end{array}$$

Thus, our argument was indeed valid!

Example: logic programming There are some langages based on logic, for example *Prolog*.
We have predicates, such as `lecturer(L, C)` (at a course) and `student(S, C)`.
Then we can add facts: `lecturer(Karl, CS101)`, `student(Frank, CS101)`, `student(Cléa, CS101)`, . . .
We can also add rules: `teaches(L, S) :- lecturer(L, C), student(S, C)`.
Such a rule will be true if the lecturer and the student have the same class, `C`.
We can then make queries: `teaches(Karl, Frank)` which will say True. Or, we can also get a list using `teaches(Karl, X)`.

Tuesday 5th October 2021 — **Lecture 5 : Inference rules and the beginning of proofs**

5.2 Inference rules in propositional logic

Conjunction Using the tautology $p \land q \to p \land q$, we get the following inference rule:

$$\begin{array}{rl} & p \\ & q \\ \hline \therefore & p \land q \end{array}$$

Modus Ponens Using the tautology $(p \land p \to q) \to q$, we get the following inference rule:

$$\begin{array}{rl} & p \to q \\ & p \\ \hline \therefore & q \end{array}$$

Modus Tollens Using the tautology $(\neg q \land p \to q) \to \neg q$, we get the following inference rule:

$$\begin{array}{rl} & p \to q \\ & \neg q \\ \hline \therefore & \neg q \end{array}$$

This has to do with the contrapositive of the proposition.

Trivial inference rule If we know that $p \equiv q$, then $p \to q$ is a tautology. Thus:

$$\begin{array}{rl} & p \\ \hline \therefore & q \end{array}$$

Example Let's say that we know $p \to q$ and $\neg q$, then we can use the trivial inference rule:

$$\begin{array}{rl} & p \to q \\ & \neg q \\ & \neg q \to \neg p \\ \hline \therefore & \neg p \end{array}$$

We have thus just proven the Modus Tollens (it indeed had to do with the contrapositive).

Hypothetical syllogism Using the tautology $(p \to q \land q \to r) \to (p \to r)$, we get the following inference rule:

$$\begin{array}{rl} & p \to q \\ & q \to r \\ \hline \therefore & p \to r \end{array}$$

Terminology This inference is also sometimes called the "chain rule", or "transitivity of implication".

Usefulness Note that this inference rule is very important in maths, since we are always chaining our reasoning.

Example Let's say we know that "if I have passed AICC, I can advance to year 2 of the studies" and "if I can advance to year 2 of the studies, I can take Analysis 4".

We can thus deduce that "if I have passed AICC, then I can take Analysis 4".

Dumb Argument Prover Let's say we want to create a "Dumb Argument Prover", DAB *(j'effectue le DAB)*. We can use the following procedure:

1. Build the complete truth table for all premises and the conclusion.
2. Remove all rows where at least one premise is false.
3. Check whether the column of the conclusion contains only T.

The problem of this method is that if we have many premises then our truth table will be huge. Using implications help to go much faster.

Example If we have $p_1, \ldots, p_n$ and $(p_1 \land \ldots \land p_n) \to q$, then we know q.

The dumb prover would take a lot of time to build the table and remove what it does not need: it would do a lot of "unncessary work". Indeed, we would only need to check for the first line — the one where every premise is true — but the dumb prover would check every line. Anyhow, the reasoning we are developing here is much more efficient.

Resolution Using the tautology $((\neg p \lor r) \land (p \lor q)) \to (q \lor r)$, we get the following rule of inference:

$$\begin{array}{rl} & \neg p \lor r \\ & p \lor q \\ \hline \therefore & q \lor r \end{array}$$

Usefulness Resolution plays an important role in automated theorem proofing and AI. It allows to eliminate proposition variables from the premises, so it is very powerful.

Example Let's say we know that "the weather is bad or I am at the beach" and "the weather is nice or I am at home".

We can then deduce that "I am at home or at the beach".

"Resolution rules them all" "I think you have all seen the movie", but it's not a movie, it's a book ☹ (which I haven't read, but that's not the question).

Let's say we have $p \to q$ and $q \to r$. We can do the following reasoning to show that resolution includes the chain rule:

$$\begin{array}{rll} & p \to q & \text{premise} \\ & q \to r & \text{premise} \\ & \neg p \lor q & \text{equivalence} \\ & \neg q \lor r & \text{equivalence} \\ & \neg p \lor r & \text{resolution} \\ \hline \therefore & p \to r & \text{equivalence} \end{array}$$

Similarly, we can see that it it also includes Modus Ponens:

$$\begin{array}{lll} & p \to q & \text{premise} \\ & p & \text{premise} \\ & \neg p \lor q & \text{equivalence} \\ & p \lor F & \text{equivalence} \\ & q \lor F & \text{resolution} \\ \hline \therefore & q & \text{equivalence} \end{array}$$

On a similar note, we have already showed that Modus Ponens implies Modus Tollens.

Summary

Deduction	Corresponding tautology	Name
$\begin{array}{ll} & p \\ & q \\ \hline \therefore & p \land q \end{array}$	$p \land q \to p \land q$	Conjunction
$\begin{array}{ll} & p \to q \\ & p \\ \hline \therefore & q \end{array}$	$(p \land p \to q) \to q$	Modus Ponens
$\begin{array}{ll} & p \to q \\ & \neg q \\ \hline \therefore & \neg p \end{array}$	$(\neg q \land p \to q) \to \neg p$	Modus Tollens
$\begin{array}{ll} & p \to q \\ & q \to r \\ \hline \therefore & p \to r \end{array}$	$(p \to q \land q \to r) \to (p \to r)$	Hypothetical Syllogism
$\begin{array}{ll} & \neg p \lor r \\ & p \lor q \\ \hline \therefore & q \lor r \end{array}$	$((\neg p \lor r) \land (p \lor q)) \to (q \lor r)$	Resolution

Other inference rule

We can also derive the following inference rule. We give them names, but they are directly implied by the other inference rules:

Deduction	Corresponding tautology	Name
$\begin{array}{ll} & p \lor q \\ & \neg p \\ \hline \therefore & q \end{array}$	$((p \lor q) \land \neg p) \to q$	Disjunctive syllogism
$\begin{array}{ll} & p \\ \hline \therefore & p \lor q \end{array}$	$p \to (p \lor q)$	Addition
$\begin{array}{ll} & p \land q \\ \hline \therefore & p \end{array}$	$(p \land q) \to p$	Simplification

Note that the disjunctive syllogism is a simpler form of resolution, addition is dual to conjunction and simplification is a simpler form of Modus Ponens.

Warning

Note that even seemingly obvious conclusions need an argument. For example:

$$\begin{array}{lll} & p \land (p \to q) & \text{Premise} \\ & p & \text{Simplification} \\ & p \to q & \text{Simplification} \\ \hline \therefore & q & \text{Modus Ponens} \end{array}$$

Fallacies

Some people use fallacies in their argumentation (consciously or unconsciously).

Fallacy of affirming the conclusion

Note that $(p \to q \land q) \to p$ is not a tautology. Thus, the following argumentation is wrong.

"If you do every problem in this book, then you will learn discrete mathematics." You learnt discrete mathematics, thus you did every problem in this book (no).

Fallacy of denying the hypothesis

Note that $(p \to q \land \neg p) \to \neg q$ is not a tautology. Thus, the following argumentation is wrong.

"If you do every problem in this book, then you will learn discrete mathematics." You did not do every problem in this book, thus you did not learn discrete mathematics (no).

5.3 Inference rules in predicate logic

Introduction Arguments in predicate logic either use arguments from propositional logic, or rules of inference for quantified statements.

Universal Instantiation (UI) The following rule of inference is valid:

$$\begin{array}{rl} & \forall x P(x) \\ \hline \therefore & P(c) \text{ for an arbitrary } c \end{array}$$

Example

Let's say the domain consists of only Men, and Socrates is a Man. If we know that "all men are mortal", then we can conclude that "Socrates is mortal".

Now, if we have any universe of discourse (such as all beings), then the complete solution is given by:

$$\begin{array}{rll} & \forall x(\text{Man}(x) \to \text{Mortal}(x)) & \text{Premise} \\ & \text{Man(Socrates)} \to \text{Mortal(Socrates)} & \text{Universal Instantiation} \\ & \text{Man(Socrates)} & \text{Premise} \\ \hline \therefore & \text{Mortal(Socrates)} & \text{Modus Ponens} \end{array}$$

Universal Generalisation (UG) Universal generalisation is the following inference rule:

$$\begin{array}{rl} & P(c) \text{ for an arbitrary } c \\ \hline \therefore & \forall x P(x) \end{array}$$

Note that we must not do any assumption about c.

Example

This is often used implicitly in Mathematical Proofs.

For example, when computing limits with the definition, we say: "Let $\varepsilon > 0$ be given [. . .]. Therefore, we know that it is true $\forall \varepsilon > 0$."

Existential Instantiation (EI) The following rule of inference is valid:

$$\begin{array}{rl} & \exists x P(x) \\ \hline \therefore & P(c) \text{ for some element } c \end{array}$$

Example

Let's say we know that there is someone who knows Java in the class. Let's call this person a, and say that a knows Java.

Existential Generalisation (EG) Existential Generalisation is the following inference rule:

$$\begin{array}{rl} & P(c) \text{ for some element } c \\ \hline \therefore & \exists x P(x) \end{array}$$

> *Example* Let's say we know that Michelle — who is in the class — knows Java. Thus, someone knows Java in the class.

Universal Modus Ponens The following rule of inference is valid:

$$\begin{array}{rl} & \forall x (P(x) \to Q(x)) \\ & P(c) \text{ for an arbitrary } c \\ \hline \therefore & Q(c) \end{array}$$

> *Proof* This inference rule combines both the Universal Instantiation and Modus Ponens into one rule:
>
> $$\begin{array}{rll} & \forall x (P(x) \to Q(x)) & \text{Premise} \\ & P(c) \text{ for an arbitrary } c & \text{Premise} \\ & P(c) \to Q(c) & \text{Universal Instantiation} \\ \hline \therefore & Q(c) & \text{Modus Ponens} \end{array}$$
>
> This could have been used for Socrates example.

Summary

Deduction	**Name**	**Shorten name**
$\begin{array}{rl} & \forall x\ P(x) \\ \hline \therefore & P(c) \text{ for an arbitrary } c \end{array}$	Universal Instantiation	UI
$\begin{array}{rl} & P(c) \text{ for an arbitrary } c \\ \hline \therefore & \forall x\ P(x) \end{array}$	Universal Generalization	UG
$\begin{array}{rl} & \forall x (P(x) \to Q(x)) \\ & P(c) \text{ for an arbitrary } c \\ \hline \therefore & Q(c) \end{array}$	Universal Modus Ponens	
$\begin{array}{rl} & \exists x\ P(x) \\ \hline \therefore & P(c) \text{ for some element } c \end{array}$	Existential Instantiation	EI
$\begin{array}{rl} & P(c) \text{ for some element } c \\ \hline \therefore & \exists x\ P(x) \end{array}$	Existential Generalization	EG

Example We want to show that "a student in this class has not read the book" and "everyone in this class passed the first exam" imply "someone who passed the first exam has not read the book".

Let's first define predicates: Let $C(x) :=$ "x is a student in the class", $B(x) :=$ "x read the book", and $E(x) :=$ "x passed the exam". Second, let's translate the premises: We know that $\exists x (C(x) \land \neg B(x))$, and $\forall x (C(x) \to E(x))$. Third, let us also translate what we want to show: $\exists x (E(x) \land \neg B(x))$.

From then, we can do our proof:

$$\begin{array}{rll} 1 & \exists x (C(x) \land \neg B(x)) & \text{Premise} \\ 2 & \forall x (C(x) \to E(x)) & \text{Premise} \\ 3 & C(a) \to E(a) \text{ for all a} & \text{UI from (2)} \\ 4 & C(\hat{a}) \land \neg B(\hat{a}) \text{ for some } \hat{a} & \text{EI from (1)} \\ 5 & C(\hat{a}) \to E(\hat{a}) & \text{(3) is true for all } a \text{, so it is true for } \hat{a} \\ 6 & C(\hat{a}) & \text{Simplification from (4)} \\ 7 & E(\hat{a}) & \text{Modus Ponens from (5) and (6)} \\ 8 & \neg B(\hat{a}) & \text{Simplification from (4)} \\ 9 & E(\hat{a}) \land \neg B(\hat{a}) & \text{Conjunction from (7) and (8)} \\ \hline \therefore & \exists x (E(x) \land \neg B(x)) & \text{EG from (9)} \end{array}$$

5.4 Proofs

Introduction “Mathematicians prove things, that’s what they do.” (Prof. Aberer)

Apart from being at the core of mathematics, proofs have many practical applications as well. For example, it allows us to verify that a computer program is correct, to establish that operating systems are secure, to enable programs to make inferences in artificial intelligence, and so on.

Terminology A **mathematical proof** is a valid argument that establishes the truth of a statement, in particular of a theorem. A **theorem** is a statement that can be shown to be true using definitions, axioms, other theorems or rules of inference. An **axiom** is a statement which is given as true.

A **lemma** is a “helping theorem”, it is something we prove in order to prove a theorem right after. A **corollary** is a result which follows directly and trivially from a theorem. **Propositions** are less important theorems.

A **conjecture** is a statement that is being proposed to be true; we do not have a proof for it, but we think it may be true (but it may end up wrong). For example, Fermat’s last theorem was conjectured in 1634 (well not exactly, this is Fermat, he said he had a proof but that his margins were too small) and was proven in 1995. On a similar note, Euler’s conjecture (which tries to generalise Fermat’s last theorem), which was conjectured in 1769, was proven to be wrong in 1966 with a counter-example.

Informal proofs, which are generally shorter, often use multiple rules of inferences at each step, skip steps, do not state every inference rule explicitly, and so on. They are much easier to understand and to explain to people, but they make it also easier to introduce errors. This is not the way a computer would do a proof.

Theorems Many theorems assert that a property holds for all elements in a domain, such as the integers, the real numbers, or discrete structures. Thus, they are often of the form:

$$\forall x (P(x) \to Q(x))$$

To prove a theorem of this form, we show that:

$$P(c) \to Q(c)$$

where c is an arbitrary element of the domain. We can then use the universal generalisation to get the theorem.

Note that, often, the universal quantifier is omitted by standard mathematical convention. For example “if $x > y$, where x and y are positive real numbers, then $x^2 > y^2$” really means “for all positive real numbers x and y, if $x > y$, then $x^2 > y^2$”:

$$\forall x > 0 \forall y > 0 \big(x > y \to x^2 > y^2\big)$$

Types of proof

Trivial proof	If we know that q is true, then $p \to q$ is true as well.	"If it is raining, then $1 = 1$"
Vacuous proof	If we know that p is false, then $p \to q$ is true.	"If I am both rich and poor, then $2 + 2 = 5$"
Direct proof	Assume that p is true, then use definitions, axioms and theorems together with rules of inference till the statement q results.	This is when we usually use syllogisms.
Proof by contraposition	Assume that $\neg q$ is true, then use definitions, axioms and theorems together with rules of inference till the statement $\neg p$ results.	Uses $p \to q \equiv \neg q \to \neg p$
Proof by contradiction	Assume that p and $\neg q$ are true, then perform a direct proof to construct a contradiction.	Uses $p \to q \equiv (p \land \neg q) \to F$

The two last proofs are called **indirect proofs**.

5.5 Proofs examples

Definition: parity

The integer n is **even** if there exists an integer k such that:

$$n = 2k$$

The integer n is **odd** if there exists an integer k such that:

$$n = 2k + 1$$

Note We can prove that a number is either even xor odd.

Theorem: odd squares

If n is an odd integer, then n^2 is odd.

Proof Let n be an odd number. Therefore, there exists k, such that $n = 2k + 1$ (by definition). Thus, we have:

$$n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2\left(2k^2 + 2k\right) + 1$$

Let $k' = 2k^2 + 2k$. We know it's an integer (we'd have to prove that as well to be really formal). Therefore,

$$n^2 = 2k' + 1$$

Which means that n^2 is odd.

□

Wednesday 6th October 2021 — **Lecture 6 : The end of proofs and the beginning of sets**

Definition: Rational number

The real number r is **rational** if there exist integers p and q, with $q \neq 0$, such that:

$$r = \frac{p}{q}$$

Theorem: Rational sum

The sum of two rational number is rational.

Proof Let r_1, r_2 be rational numbers. Therefore, $r_1 = \frac{p_1}{q_1}$ and $r_2 = \frac{p_2}{q_2}$ where p_1, q_1, p_2, q_2 are integers, and $q_1, q_2 \neq 0$. So:

$$r_1 + r_2 = \frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1 q_2 + p_2 q_1}{q_1 q_2}$$

Let $p' = p_1 q_2 + p_2 q_1 \in \mathbb{N}$ and $q' = q_1 q_2 \in \mathbb{N}$. Since both q_1 and q_2 are non zero, then q' is not as well. So,

$$r_1 + r_2 = \frac{p'}{q'}$$

which is rational.

□

Theorem (proof by contraposition) Let n be an integer. If $3n + 2$ us odd, then n is odd.

Proof Let's do a proof by contraposition. The contrapositive of our sentence — thus what we are trying to show — is "if n is even (not odd), then $3n + 2$ is even (not odd)."

Let n be even, meaning there exists a k such that $n = 2k$ (by definition). So,

$$3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1)$$

By letting $k' = 3k + 1$, we have $3n + 2 = 2k'$, so $3n + 2$ is even.

□

Theorem (proof by contraposition) Let n be an integer. If n^2 is odd, then n is odd.

Proof Again, let's prove this theorem by contraposition. The contrapositive of our sentence is "if n is even, then n^2 is even".

Let n be an even number, i.e. $n = 2k$ for some integer k. So,

$$n^2 = (2k)^2 = 4k^2 = 2\overbrace{\left(2k^2\right)}^{k'} = 2k'$$

So, $n^2 = 2k'$, which means it is even.

□

Pigeon-hole principle (proof by contradiction) If more than N items are distributed in any manner over N bins, there must be a bin containing at least two items.

Remark We will come back much later to this theorem; we will use it in counting.

Proof The proposition p is "more than N items are distributed over N bins", and q is "one bin contains at least 2 items". Thus, the negation of q — $\neg q$ — is "all bins contain at most 1 item".

Since it is a proof by contradiction, we suppose p and $\neg q$. $\neg q$ implies that there are at most N items, but it is a contradiction with p. In other words, $\neg p \wedge p \equiv F$.

□

Contraposition versus contradiction We can realise that the last proof could also have been done with a proof by contraposition.

More generally, any proof by contradiction can be formulated as a proof by contradiction, but the reciprocal is not always true. Indeed, for a proof by contraposition, we show that:

$$\neg q \to \neg p$$

Thus, if we have a direct proof for $\neg q \to \neg p$, we can also do a proof by contradiction. Indeed, since we assumed $\neg q$, we get $\neg p$, which is a contradiction when put together with out other assumption, p:

$$(p \land \neg q) \to (p \land \neg p)$$

Generally, proofs by contradiction are stronger, since they allow us to show that:

$$(p \land \neg q) \to (r \land \neg r)$$

Theorem (genuine proof by contradiction) $\sqrt{2}$ is irrational.

Proof Let's suppose for contradiction that $\sqrt{2}$ is rational ($\neg q$).

So, it means that there exist integers a, b, $b \neq 0$ such that $\sqrt{2} = \frac{a}{b}$ where a and b have no common factors (r). Then,

$$2 = \frac{a^2}{b^2} \implies 2b^2 = a^2 \implies a^2 \text{ even } \implies a \text{ even}$$

We can thus write $a = 2c$. So,

$$2b^2 = a^2 = 4c^2 \implies b^2 = 2c^2 \implies b^2 \text{ even} \implies b \text{ even}$$

Therefore, we can write $b = 2d$. However, since a and b are both even, it implies that have a common factor, 2 ($\neg r$). We thus have our contradiction.

□

5.6 Other proof methods

Proof for Biconditional Statements To prove a theorem which is a biconditional statement, i.e of the form $p \leftrightarrow q$, we show that $p \to q$ and $q \to p$ are both true.

This is based on:

$$p \leftrightarrow q \equiv p \to q \land q \to p$$

Example Let's say we want to show that, with n an integer, n is odd if and only if n^2 is odd. To do that, we would need that n is odd implies that n^2 is odd, and that if n^2 is odd then n is odd. Since we have already proven both, we have proven our theorem.

□

Proof by cases We can see the following equivalence:

$$(p_1 \lor \ldots \lor p_n) \to q \equiv (p_1 \to q) \land \ldots \land (p_n \to q)$$

Thus, to prove a theorem of the form $(p_1 \lor \ldots \lor p_n) \to q$, we only have to prove every $p_i \to q$ — called a case — separately.

Example

Let's say we want to show that if n is an integer, then we have $n^2 \geq n$.

Let's check when $n = 0$:

$$0^2 \geq 0 \implies n^2 \geq n$$

When $n \geq 1$:

$$n \geq 1 \xRightarrow{\cdot n} n^2 \geq n$$

When $n \leq -1$:

$$n \leq -1 \leq 0 \text{ and } n^2 > 0 \implies n^2 > 0 \geq n$$

□

WLOG

In the context of a proof by case, there may be a case that follows trivially from another (for example, by swapping roles of variables). In such a scenario, we can use the word "WLOG" — "Without Loss Of Generality". Note that in French, we say "SPDG" — "Sans Perte De Généralité".

For example, let's say we want to show that if x, y are integers and both xy and $x + y$ are even, then both x and y are even. If we want to prove this problem by contrapositive, we will have x or y is odd. We can take WLOG that x is odd. Indeed, if y is odd the problem is completely symmetrical, and the proof would be the exact same.

Proof by counterexample

To establish that $\forall x P(x)$ is false, i.e. that $\neg\forall x P(x)$ is true, we only need to find a c such that $\neg P(c)$ is true. Indeed:

$$\neg\forall x P(x) = \exists x \neg P(x)$$

In this case, c is called a **counterexample** to the assertion $\forall x P(x)$.

Example

Let's say we want to show that "every positive integer is the sum of the square of 2 integers" is false.

We notice that $1 = 1^2 + 0^2$, $2 = 1^2 + 1^2$. However, $3 = 3 + 0 = 2 + 1$, so 3 is a counterexample.

Existence proof

To show that $\exists x P(x)$, we have two possibilities:

- **Constructive Proof:** Find a c such that $P(c)$ is true.
- **Non-constructive proof:** Find c_1, c_2 such that $P(c_1) \vee P(c_2)$ is true. Indeed:

$$(P(c_1) \to \exists x P(x)) \wedge (P(c_2) \to \exists x P(x)) \equiv (P(c_1) \vee P(c_2)) \to \exists x P(x)$$

This is called "non-constructive", because we do not know if it is c_1 or c_2 which satisfies this property.

Example of constructive proof

There exists a positive integer that can be written as a sub of cubes in two ways:

$$1729 = 10^3 + 9^3 = 12^3 + 1^3$$

Note that I knew this number and impressed the professor and the whole class! ☺This is the Hardy-Ramanujan constant, and it has a great history.

Example of non-constructive proof We want to show that there exists irrational numbers x and y such that x^y is rational.

Let's consider $\sqrt{2}^{\sqrt{2}}$. If it is rational, then $x_1 = \sqrt{2}$ and $y_1 = \sqrt{2}$ are such that $x_1^{y_1}$ is rational. Else, if $\sqrt{2}^{\sqrt{2}}$ is irrational, we have:

$$\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$$

In other words, in this second case, we have found $x_2 = \sqrt{2}^{\sqrt{2}}$, $y_2 = \sqrt{2}$, which are both irrational, such that $x_2^{y_2}$ is rational.

□

5.7 Misteaks in proofs

Example 1 Let's consider the following proof:

$$\begin{aligned} -1 &= (-1)^1 && \textit{Step 1} \\ &= (-1)^{\frac{2}{2}} && \textit{Step 2} \\ &= \left((-1)^2\right)^{\frac{1}{2}} && \textit{Step 3} \\ &= 1^{\frac{1}{2}} && \textit{Step 4} \\ &= 1 && \textit{Step 5} \end{aligned}$$

This proof is definitely wrong since $-1 \neq 1$. We can see that, actually, in the third step, we are abusing the following law:

$$\forall x \geq 0, \quad (x^a)^b = x^{ab}$$

Example 2 We are looking for a solution of $\sqrt{2x^2 - 1} = x$:

$$\begin{aligned} & \sqrt{2x^2-1} = x && \\ \iff & 2x^2 - 1 = x^2 && \textit{Step 1} \\ \iff & x^2 - 1 = 0 && \textit{Step 2} \\ \iff & (x-1)(x+1) = 0 && \textit{Step 3} \\ \iff & x = 1 \lor x = -1 && \textit{Step 4} \end{aligned}$$

However, $x = -1$ is not a solution, so there is a mistake in our reasoning.
The error is the first step. Indeed, it is true that $\forall x \forall y \left(x = y \to x^2 = y^2\right)$, but not $\forall x \forall y \left(x = y \leftarrow x^2 = y^2\right)$. We have thus introduced an extraneous solution.

Example 3 Let's consider the following reasoning:

$(p \to q) \lor (q \to p)$ is a tautology.	*Step 1*
Let $p :=$ "n is odd" and $q :=$ "n is prime".	*Step 2*
However, neither (odd → prime) nor (prime → odd) is true.	*Step 3*

There is a contradiction, so this reasoning must be wrong.
Here, we are abusing the distribution of quantifier. Indeed, we know that the following proposition is a tautology:

$$\forall n (p(n) \to q(n) \lor q(n) \to p(n))$$

And we are saying that this is a contradiction since the following proposition is wrong:

$$\forall n (p(n) \to q(n)) \lor \forall n (q(n) \to p(n))$$

However, those two propositions are not equal; we cannot distribute the universal quantifier.

Chapter 6

Sets and functions

6.1 Introduction to sets

Short History

Most of the set theory was developed between the nineteenth and the twentieth century.
Cantor, the founder of this theory, discovered uncountable sets. Peano introduces notations (such as $\in$, $\cup$, $\cap$) and axioms for natural numbers. Zermelo and Fraenkel developed the currently widely accepted axioms for set theory. Bertrand Russell (known for his participation to the CND) wrote Principia Mathematica, deriving all mathematics from primitive axioms; he is known for his antimony on set theory.

Introduction

Sets are one of the basic building blocks in discrete mathematics. They are the basis for counting, functions, relations, and so on. Many programming language have set operations, and databases are sets of discrete objects.
Set theory is an important branch of mathematics. Many different systems of axioms have been used to develop set theory. In our context, we are not interested by formal set of axioms, we will use what is called **naïve set theory**.
Note that sets are basic and important abstractions in mathematics and computer science. Indeed, many data structures are constructed from the abstractions we will develop. For example, unordered collections are similar to sets, ordered collections are similar to sequences, network and graphs are similar to relations, databases are similar to relations, and so on. Moreover, many computing models are made using these abstractions. For example, finite state machines are similar to relations, programs are decomposed into functions, and costs of programs are expressed as functions.

Definition

A **set** is an unordered collection of objects. The objects in a set are called the **elements** of the set. A set is said to **contain** its elements.
The notation $a \in A$ denotes that mka is an element of the set A. If a is not an element of A, we write $a \notin A \equiv \neg(a \in A)$

Roster method

To describe a set, we can use what is called the **Roster method**. Basically, we list all elements:

$$S = \{a, b, c, d\}$$

We are basically saying

$$\forall x (x \in S \leftrightarrow (x = a \vee x = b \vee x = c \vee x = d))$$

If we can continue the set in a logical way, we can use ellipses:

$$T = \{0, 1, 2, 3, 4, \ldots\}$$

Order

> Note that sets are unordered, so the order is not important:
>
> $$S = \{a, b, c, d\} \equiv \{b, c, a, d\}$$

Example For example, the set of all vowels in the English alphabet is:

$$\{a, e, i, o, u, y\}$$

The set of all odd positive integers less than 10 is:

$$\{1, 3, 5, 7, 9\}$$

The set of all positive integers less than 100 is:

$$\{1, 2, \ldots, 99\}$$

The set of all integers less than 0 is:

$$\{-1, -2, \ldots\}$$

Sets of number The following sets of numbers are really important:

$\mathbb{N}$	Natural numbers	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	Integers	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{Z}_+$	Positive integers	$\{1, 2, 3, \ldots\}$
$\mathbb{Q}$	Rational numbers	
$\mathbb{R}$	Real numbers	
$\mathbb{R}_+$	Positive real numbers	
$\mathbb{C}$	Complex numbers	

Note that ***0 is NOT positive, this is really important for tests!***

Set-builder notation We can specify the property or properties that all members satisfy:

$$S = \{x | P(x)\}$$

$P(x)$ might be expressed in natural language or predicate logic.

> *Examples* We can construct the following sets:
>
> $$S = \{x | x \text{ is a positive integer less than } 100\}$$
>
> $$\mathbb{Q} = \left\{x \in \mathbb{R} | \exists p, q \in \mathbb{N} \text{ such that } x = \frac{p}{q}\right\}$$

Interval notation For sets of number we can use the following notations:

$$[a, b] = \{x | a \leq x \leq b\}, \quad [a, b) = \{x | a \leq x < b\}$$

$$(a, b] = \{x | a < x \leq b\}, \quad (a, b) = \{x | a < x < b\}$$

Note that we call $[a, b]$ a **closed interval**, and (a, b) an **open interval**. Sometimes, we also note:

$$(a, b) =]a, b[$$

Important sets The **universal set**, U, is the set containing everything currently under consideration. It depends on the context. Note that it is very dangerous, because it is often implicit. The **empty set** is the set with no element. It can be noted as $\varnothing$ or $\{\}$.

> *Important remark* Note that sets can e elements of sets. For example, the following set is valid:
>
> $$\{\{1, 2, 3\}, a, \{b, c\}\}$$

Thus, the empty set is different from a set containing the empty set:

$$\varnothing \neq \{\varnothing\}$$

Tuesday 12th October 2021 — **Lecture 7 : The end of sets and the beginning of functions**

Definition: Set equality Two sets A and B are equal, written $A = B$, if and only if A and B have the same elements. In other words:

$$A = B \equiv \forall x (x \in A \leftrightarrow x \in B)$$

Definition: Subsets The set A is as **subset** of B, written $A \subseteq B$, if and only if every element of A is also an element of B. In other words:

$$A \subseteq B \equiv \forall x (x \in A \rightarrow x \in B)$$

We say that A is a **proper subset** of B, written $A \subset B$, if and only if $A \subseteq B$ but $A \neq B$; i.e:

$$\forall x (x \in A \rightarrow x \in B) \land \exists x (x \in B \land x \notin A)$$

With important sets Note that, for any set S:

$$\varnothing \subseteq S, \quad S \subseteq S$$

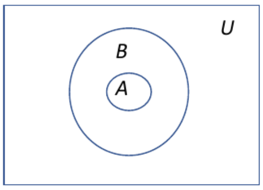
Showing this property To show that A is a subset of B, we need to show that if x belongs to A, then x also belongs to B. To show that A is not a subset of B, we need to find an element x such that $x \in A$ but $x \notin B$. To show that A is a proper subset of B, we need to show that a is a subset of B, and find an element x such that $x \in B$ and $x \notin A$.

Showing equality We can note that:

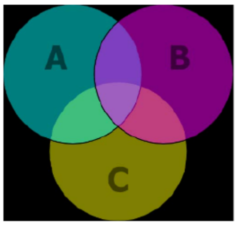
$$A = B \equiv \forall x (x \in A \leftrightarrow x \in B) \equiv \forall x (x \in A \rightarrow x \in B \land x \in B \rightarrow x \in A) \equiv A \subseteq B \land B \subseteq A$$

So, proving that two sets consists in proving that one set is included in the other, and vice-versa.

Venn Diagrams Venn Diagrams are pictures of sets, drawn as subsets of some universal set U. They are very useful for understanding, but can never be used for proofs.



$A \subseteq B$



List 1
List 2
List 3
List 4
0
4
1
3
2
5
2
0
7
0
0
0
0
0
2

6.2 Constructing sets

Power sets

The set of all subsets of a set A, written $\mathcal{P}(A)$, is called the **power set** of A.

> *Example* If $A = \{a, b\}$, then:
>
> $$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$$

> *Remark* Note that, for any set A, then:;
>
> $$\emptyset \subseteq \mathcal{P}(A) \quad \text{and} \quad A \subseteq \mathcal{P}(A)$$

Tuples

The **ordered n-tuples** $(a_1, \ldots, a_n)$ is the ordered collections that has a_1 as its first element, a_2 as its second elements, and so on until a_n, which is its last element.
2-tuples are called **ordered pairs**.

> *Remarks* We can have multiple times the same element in a tuple. For example:
>
> $$(1, 1, 1) \neq (1, 1) \neq (1)$$
>
> Concerning ordered pairs, the “ordered” is important. Indeed, in general:
>
> $$(a, b) \neq (b, a)$$

> *Equality* Two n-tuples are equal if and only if their corresponding elements are equal:
>
> $$(a_1, \ldots, a_n) = (b_1, \ldots, b_n) \iff a_1 = b_1 \land \ldots \land a_n = b_n$$

Cartesian product

The **Cartesian product** of two sets A and B, denoted by $A \times B$, is the set of ordered pairs (a, b) where $a \in A$ and $b \in B$:

$$A \times B = \{(a, b) | a \in A \land b \in B\}$$

A subset R of the Cartesian product $A \times B$ is called a **relation** from the set A to the set B.
The **Cartesian products** of the sets $A_1, \ldots, A_n$, denoted $A_1 \times \ldots \times A_n$ is the set of the ordered n-tuples $(a_1, \ldots, a_n)$, where a_i belongs to A_i:

$$A_1 \times \ldots \times A_n = \{(a_1, \ldots, a_n) | a_i \in A_i\}$$

> *Remark* We can note that, in general, it is not commutative:
>
> $$A \times B \neq B \times A$$
>
> Note that, in general, it is not associative either:
>
> $$A \times (B \times C) \neq (A \times B) \times C$$

> *Example* Let $A = \{a, b\}$ and $B = \{1, 2, 3\}$. Then:
>
> $$A \times B = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$$
>
> A relation is any subset, so, for example:
>
> $$R = \{(a, 2), (b, 1)\}$$
>
> Moreover, if $A = \{0, 1\}$, $B = \{1, 2\}$ and $C = \{0, 1, 2\}$, then:
>
> $$A \times B \times C = \{(0, 1, 0), (0, 1, 1), (0, 1, 2), (0, 2, 0), \ldots, (1, 2, 2)\}$$

Truth set Given a predicate P and a domain D, we define the **truth set** of P to be the set of elements in D for which $P(x)$ is true. This is denoted by:

$$\{x \in D | P(x)\}$$

> *Example* The truth set of $P(x)$ where $P(x) := |x| = 1$ and the domain is the integers, is the set $\{-1, 1\}$.

Set cardinality If there are $n \in \mathbb{N}$ distinct elements in a set S, we say that S is **finite** (which basically mean that we could write the set down), otherwise it is **infinite**.
The **cardinality** of a finite set S, denoted $|S|$, is the number of (distinct) elements of S.

> *Examples* We see the following properties:
>
> $$|A| = n \implies |\mathcal{P}(A)| = 2^n$$
>
> $$|A| = n, |B| = m \implies |A \times B| = n \cdot m$$
>
> $$|\emptyset| = 0, \quad |\{\emptyset\}| = 1$$
>
> Finally, for example, the set of integers is infinite.

Example The power set of the empty set can be sometimes a bit tricky. For example:

- $\mathcal{P}(\emptyset) = \{\emptyset\}$
- $\mathcal{P}(\mathcal{P}(\emptyset)) = \mathcal{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$
- $\mathcal{P}(\mathcal{P}(\{\emptyset\})) = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$

We can make the following conjecture:

$$|\mathcal{P}^n(\emptyset)| = 2^n$$

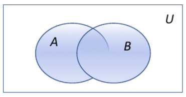
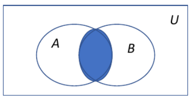
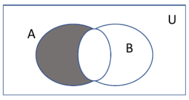
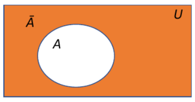
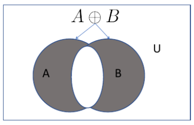
Where $P^n(A)$ is the repeated power set over the set A.

Proposition If $\mathcal{P}(A) = \mathcal{P}(B)$, then $A = B$.

> *Proof* Let's do a proof by contraposition. Since $A \neq B$, then there exists an element x which is in a set but not in the other. Let's say wlog that $x \in A$ and $x \notin B$.
> So:
>
> $$\{x\} \subset \mathcal{P}(A), \quad \{x\} \notin \mathcal{P}(B)$$
>
> Thus, $\mathcal{P}(A) \neq \mathcal{P}(B)$.
>
> □

6.3 Set operations

Set operations

Name	Definition	Example	Venn Diagram
Union	$\{x \mid x \in A \vee x \in B\}$	$\{1,2\} \cup \{2,3\} = \{1,2,3\}$	
Intersection	$\{x \mid x \in A \wedge x \in B\}$	$\{1,2\} \cap \{2,3\} = \{2\}$	
Difference	$\{x \mid x \in A \wedge x \notin B\}$	$\{1,2,3\} \setminus \{2,4\} = \{1,2\}$	
Complement	$\{x \in U \mid x \notin A\}$	$\overline{(-\infty,3]} = (3,+\infty)$	
Symmetric Difference	$(A \setminus B) \cup (B \setminus A)$	$\{1,2,3\} \oplus \{2,3,4\} = \{1,4\}$	

Name	Analogy with propositional connectives	Comment
Union	$p \vee q$	
Intersection	$p \wedge q$	If $A \cap B = \emptyset$ then the sets are said to be **disjoint**.
Difference		We can see the following property: $$A \setminus B = A \cap \overline{B}$$ The difference of A and B is also called the complement of B with respect to A
Complement	$\neg p$	We can see the following property: $$\overline{A} = U \setminus A$$ The complement of A is also denoted by A^C.
Symmetric Difference	$p \oplus q$	

Cardinality of set Union

The principle of Inclusion-Exclusion tells us that:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

Indeed, we would count twice $|A \cap B|$ else. This result will be very useful later in the course, in counting.

6.4 Set identities

Set identities Set identities can be understood as analogues of logical equivalences in propositional logic.
For example, we have De Morgan Laws for sets:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}, \quad \overline{A \cup B} = \overline{A} \cap \overline{B}$$

To prove such identities, we have different ways. The first one is to use set-builder notation and propositional logic. The second one is to prove that each set is a subset of the other (as mentioned earlier). The third one is to do a membership table (this is an analogue of truth tables).

Example: set-builder notation Let's use the set-builder notation to prove that:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

This can be done using the following steps:

$$\begin{aligned}\overline{A \cap B} &= \{x | x \notin A \cap B\} \\ &= \{x | \neg x \in A \cap B\} \\ &= \{x | \neg (x \in A \land x \in B)\} \\ &= \{x | \neg x \in A \lor \neg x \in B\} \\ &= \{x | x \notin A \lor x \notin B\} \\ &= \left\{x | x \in \overline{A} \lor x \in \overline{B}\right\} \\ &= \left\{x | x \in \overline{A} \cup \overline{B}\right\} \\ &= \overline{A} \cup \overline{B}\end{aligned}$$

Example: membership table Let's use a membership table to prove that:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

We can draw the following table:

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cup \overline{B}$	$A \cap B$	$\overline{A \cap B}$
1	1	0	0	0	1	0
1	0	0	1	1	0	1
0	1	1	0	1	0	1
0	0	1	1	1	0	1

The fifth and the seventh columns are indeed equals.

Identities We get the exact same identities with sets as the ones we had with propositional logic:

Identity	**Name**
$A \cap U = A$ $A \cup \emptyset = A$	Identity laws
$A \cup U = U$ $A \cap \emptyset = \emptyset$	Domination laws
$A \cup A = A$ $A \cap A = A$	Idempotent laws
$\overline{\overline{A}} = A$	Double negation law
$\overline{A \cap B} = \overline{A} \cup \overline{B}$ $\overline{A \cup B} = \overline{A} \cap \overline{B}$	De Morgan's Laws
$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$	Absorption laws
$A \cup \overline{A} = U$ $A \cap \overline{A} = \emptyset$	Complement laws
$A \cup B = B \cup A$ $A \cap B = B \cap A$	Commutative laws
$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$	Associative laws
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	Distributive laws

Generalised unions and intersections

Since unions and intersections are associative, we can introduce the following notations. Let $A_1, \ldots, A_n$ be sets. Then:

$$\bigcup_{i=1}^{n} A_i = A_1 \cup \ldots \cup A_n$$

$$\bigcap_{i=1}^{n} A_i = A_1 \cap \ldots \cap A_n$$

We have a similar notation for propositional logic.

Russel's Paradox

Bertrand Russel made the following paradox.
Let S be the set that contains a set x if the set x does not belong to itself. In other words:

$$S = \{x | x \notin x\}$$

If $S \in S$, then:

$$S \in \{x | x \notin x\} \implies S \notin S$$

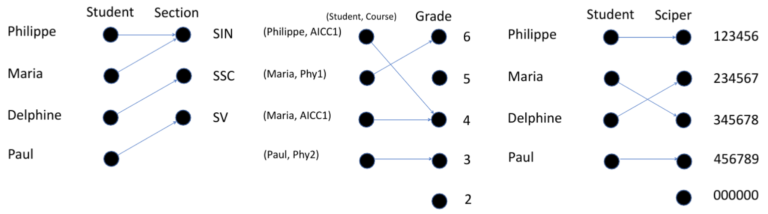
Which is a contradiction. Else, if $S \notin S$, then we have $S \in S$, which is also a contradiction.
Those kind of paradoxes led to the building of stronger axioms for set theory, making the difference from "naive set theory", in which one could build any set.

6.5 Function

Definition

Let A and B be non-empty sets. A **function** f from A to B is an assignment of exactly one element of B to each element of A, written $f : A \mapsto B$. We write $f(a) = b$ where b is the unique element of B assigned by the function to the element a of A.
Functions are sometimes called **mappings** or **transformations**.

Example



We will come back alter to this, but the first function is surjective, the second one is neither surjective nor injective, and the last one is injective.

Terminology Given a function $f : A \mapsto B$, we say that f **maps** A to B, or f is a **mapping** from A top B. A is called the **domain** of f and B is called the **codomain** of f.
If $f(a) = b$, then b is called the **image** of a under f. a is called the **preimage** of b.
The **range** of f is the set of all image of points in A under f. We denote it by $f(A)$.
More generally, we define $f(S)$, where $S \subseteq A$ as:

$$f(S) = \{f(s) | s \in S\}$$

Two functions are **equal** when they have the same domain, the same codomain, and map each element of the domain to the same element of the codomain.

Representing functions Functions may be specified in different ways. First, it can be an explicit statement of the assignment, such as a table of students and their grades. Second, it can be a formula, such as $f(x) = x + 1$. Third, it can be a computer program, such as a Python program that produces the number 2^n when given an integer n.

Definition: Injection A function f is said to be **one-to-one** or **injective** if and only if:

$$f(a) = f(b) \implies a = b$$

Such a function is said to be an injection.

Example Every Sciper number can only be assigned to one student. If sciper(Alberts) = sciper(Karen), then we have a problem.

Contrapositive Proving that a function may be easier using the contrapositive:

$$a \neq b \implies f(a) \neq f(b)$$

Definition: Surjection A function f from A to B is called **onto** or **surjective**, if and only if:

$$\forall b \in B \exists a \in A f(a) = b$$

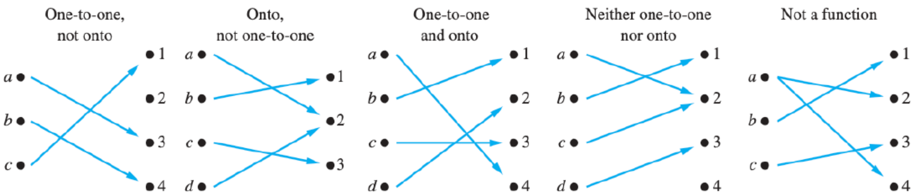
Such a function is called a **surjection**.

Example Every section has at least one student. If there is a section at EPFL which has no student, there is probably a problem.

Definition: Bijection A function f from A to B is a **one-to-one correspondance** or **bijection**, if it is both one-to-one and onto (surjective and injective).

Example Sciper numbers are bijection. Indeed, as explained before they are injective. Moreover, they are also surjective, since it would make no sense that there exists a sciper number which is assigned to no student.

Illustration



Note that the last one is not a function since a has two images.

Wednesday 13th October 2021 — Lecture 8 : End of functions and beginning of relations

Proofs

Let $f : A \mapsto B$ be a function.
To prove that f is injective, we select arbitrary $x, y \in A$, we show that if $f(x) = f(y)$ then $x = y$. To show that f is not injective, we find $x, y \in A$ such that $x \neq y$ and $f(x) = f(y)$.
To prove that f is surjective, we select an arbitrary $y \in B$, and we show the existence of an element $x \in A$ such that $f(x) = y$. To show that f is not surjective, we must find a $y \in B$ such that $f(x) \neq y$ for all $x \in A$.

Examples

$f : \mathbb{Z} \mapsto \mathbb{Z}, f(x) = x + 1$	Injective and surjective.
$f : \mathbb{N} \mapsto \mathbb{N}, f(x) = x + 1$	Injective but not surjective since $f(x) = 0$ has no solution.
$f : \mathbb{Z} \mapsto \mathbb{Z}, f(x) = x^2$	Not injective since $f(-1) = 1 = f(1)$ and not surjective since $f(x) = -1$ has no solution.

Inverse functions

Let f be a bijection from A to B. The **inverse** of f, denoted f^{-1}, is the function from B to A defined as:

$$f^{-1}(y) = x \iff f(x) = y$$

Importance of bijectivity

If f was not injective, then f^{-1} would have multiple values for some x.

If f was not surjective, the f^{-1} would not be defined for some x.

Example

The inverse of $f : \mathbb{Z} \mapsto \mathbb{Z}, f(x) = x + 1$ is:

$$\begin{aligned} f^{-1} : \mathbb{Z} &\longmapsto \mathbb{Z} \\ x &\longmapsto x - 1 \end{aligned}$$

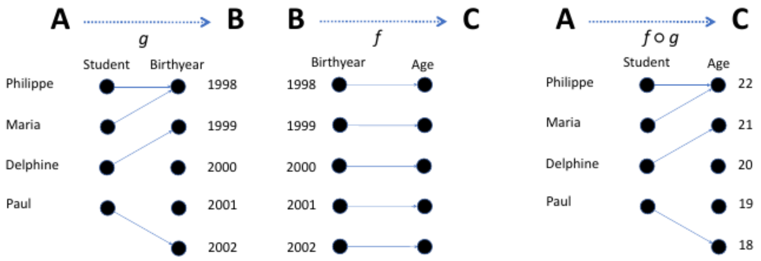
However, the function $g : \mathbb{R} \mapsto \mathbb{R}, g(x) = x^2$ has no inverse since it is neither surjective nor injective.

Composition of functions

Let $f : B \mapsto C$ and $g : A \mapsto B$ be functions. The **composition** of f with g, denoted $f \circ g$ is the function from A to C defined by:

$$(f \circ g)(x) = f(g(x))$$

Example



Similarly, if $f(x) = x^2$ and $g(x) = x + 1$, then:

$$f(g(x)) = f(x+1) = (x+1)^2 = x^2 + 2x + 1$$

$$g(f(x)) = g\left(x^2\right) = x^2 + 1$$

We can note that, in general, composition is not commutative.

Partial functions A **partial function** f from a set A to a set B is an assignment to each element a in a *subset* of A, called the **domain of definition** of f, of a unique element b in B. The sets A and B are called the **domain** and **codomain** of f respectively. We say that f is **undefined** for elements in A that are not in the domain of definition of f. When the domain of definition of f equals A, we say that f is a **total function**.

Example Let the following function:

$$\begin{aligned} f : \mathbb{Z} &\longmapsto \mathbb{R} \\ n &\longmapsto \sqrt{n} \end{aligned}$$

This is a partial function, since its -1 is in the domain but the function is undefined for it. The domain of definition of the function is $\mathbb{N}$ (f is undefined for negative integers).

Proposition Let f, g be functions. If g and $g \circ f$ are both injective, then f is injective.

Proof Let's assume for contradiction that there exists x_1, x_2 such that $f(x_1) = f(x_2) = c$ and $x_1 \neq x_2$. So,

$$(g \circ f)(x_1) = g(f(x_1)) = g(c) = g(f(x_2)) = (g \circ f)(x_2)$$

Since $g \circ f$ is injective,this implies that $x_1 = x_2$. This is a contradiction.

□

Properties Let $f : A \mapsto B$. Then for any sets $S, T \subseteq A$:

1. $f(S \cup T) = f(S) \cup f(T)$
2. $f(S \cap T) \subseteq f(S) \cap f(T)$

Justification for (2) Let's have an example of $f(S \cap T) \neq f(S) \cap f(T)$. Let $f : \{0, 1\} \mapsto \{0, 1\}$, defined such that $f(0) = 1$ and $f(1) = 1$. So, on the left hand side:

$$f(\{0\} \cap \{1\}) = f(\emptyset) = \emptyset$$

On the right hand side:

$$f(\{0\}) \cap f(\{1\}) = \{1\} \cap \{1\} = \{1\}$$

Indeed, we can write intersections the following way:

$$f(S \cap T) = \{y | \exists x(x \in S \cap T \land y = f(x))\}$$

And:

$$\begin{aligned} f(S) \cap f(T) &= \{y | \exists x(x \in S \land y = f(x))\} \cap \{y | \exists x(x \in T \land y = f(x))\} \\ &= \{y | \exists x(x \in S \land y = f(x)) \land \exists x(x \in T \land y = f(x))\} \end{aligned}$$

But to prove that both sides are equal, we would need to have the distributivity of the existence quantifier over $\land$, which is not a true law.

Example Let $f :]0,1[\mapsto \mathbb{R}$ be the function defined as

$$f(x) = \begin{cases} 2 - \dfrac{1}{x}, & x \in \left]0, \dfrac{1}{2}\right[\\ \dfrac{1}{1-x} - 2, & x \in \left[\dfrac{1}{2}, 1\right[\end{cases}$$

We can convince ourselves that it is monotonic by studying the different cases, and thus injective. This should be enough for a MCQ, but let's prove it formally.

Proof Let's prove it by contraposition, we want to show that if $x_1 \neq x_2$, then $f(x_1) \neq f(x_2)$.

Case 1: Let's pick $x_1 < \frac{1}{2}$ and $x_2 \geq \frac{1}{2}$. We then have $f(x_1) < 0$ and $f(x_2) \geq 0$. So, necessarily:

$$f(x_1) \neq f(x_2)$$

Case 2: Let's pick $x_1, x_2 < \frac{1}{2}$, with $x_1 \neq x_2$. Then we have $f(x_1) = 2 - \frac{1}{x_1}$ and $f(x_2) = 2 - \frac{1}{x_2}$. However, we know that:

$$x_2 \neq x_1 \iff \frac{1}{x_2} \neq \frac{1}{x_1} \iff 2 - \frac{1}{x_2} \neq 2 - \frac{1}{x_1}$$

Case 3: Let's pick $x_1, x_2 \geq \frac{1}{2}$, with $x_1 \neq x_2$. This is left as an exercise for the reader.

Chapter 7

Relations, sequences and summations

7.1 Relations

Definition: binary relations A **binary relation** R from a set A to a set B is a subset of their Cartesian product:

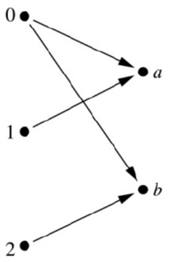
$$R \subseteq A \times B$$

We notice that $\emptyset$ is a relation from any two sets A and B.

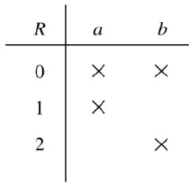
> *Example* Let $A = \{0, 1, 2\}$ and $B = \{a, b\}$. Then, the following set is a relation from A to B:
>
> $$R = \{(0, a), (0, b), (1, a), (2, b)\}$$

> *Link with functions* Relations are more general concepts than functions. Indeed, we can always construct a relation from a function:
>
> $$\{(a_1, f(a_1)), (a_2, f(a_2)), \dots\}$$
>
> In general, the inverse cannot be done (there might be a number which would get multiple images).

Representations We can use directed graphs and tables to express relations:



directed graph



R	a	b
0	×	×
1	×	
2		×

table

We see that this could not be a function, since there are multiple arrows leaving the element 0.

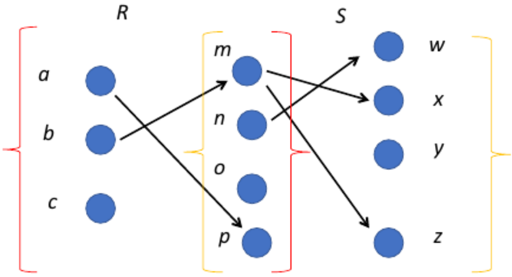
Combining relations Let R_1 and R_2 be two relations. We can combine them using basic set operations to form new relations, such as:

$$R_1 \cup R_2, \quad R_1 \cap R_2, \quad R_1 \setminus R_2, \quad R_2 \setminus R_1$$

Compositions Let A, B, C be sets, R be a relation from A to B, and S be a relation from B to C. The **composite** of R and S, denoted $S \circ R$, is the relation consisting of ordered

pairs (a, c), where $a \in A$, $c \in C$ and for which there exists an element $b \in B$ such that $(a, b) \in R$ and $(b, c) \in S$.
Note that the order is important, $S \circ R \neq R \circ S$ in general (one may even be defined, whereas the other not).

Example Let's look at the following diagram:

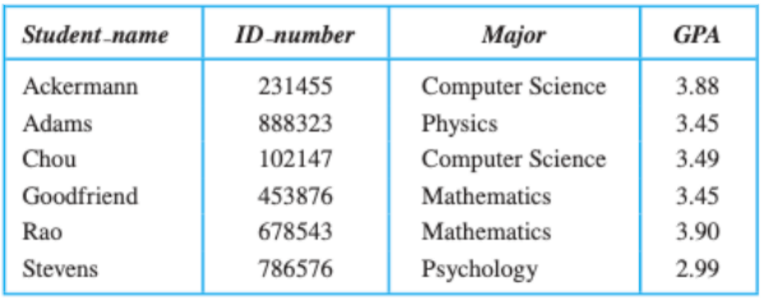


In this case, we have:

$$S \circ R = \{(b, x), (b, z)\}$$

Definition: *N*-ary relations Let $A_1, \ldots, A_n$ be sets. An ***n*-ary relation** on these sets is a subset of the Cartesian product $A_1 \times \ldots \times A_n$. The sets are called the **domains** of the relation, and n is called its **degree**.

Example Database tables are n-ary relations:



Student_name	*ID_number*	*Major*	*GPA*
Ackermann	231455	Computer Science	3.88
Adams	888323	Physics	3.45
Chou	102147	Computer Science	3.49
Goodfriend	453876	Mathematics	3.45
Rao	678543	Mathematics	3.90
Stevens	786576	Psychology	2.99

Personal remark: Note that Livai (or Mikasa) is a student in this school.

7.2 Relations on a set

Definition: Relation on a set A **binary relation *R* on a set *A*** is a subset of the Cartesian product $A \times A$. In other words, it is a relation from A to A.

Example Let $A = \{a, b, c\}$. The following set is a relation on A:

$$R = \{(a, a), (a, b), (a, c)\}$$

Definition: Reflexive relation A relation R on a set A is **reflexive** if and only if $(a, a) \in R$ for every element $a \in A$.
In other words, R is reflexive if and only if:

$$\forall x (x \in A \implies (x, x) \in R)$$

Definition: Symmetric relations A relation R on a set A is **symmetric** if and only if $(b, a) \in R$ whenever $(a, b) \in R$ for all $a, b \in A$.
In other words, R is symmetric if and only if:

$$\forall x \forall y ((x, y) \in R \rightarrow (y, x))$$

Definition: Antisymmetric relations

A relation R on a set A is **antisymmetric** if and only if $(a,b) \in R$ and $(b,a) \in R$ then $a = b$ for all $a, b \in A$.
In other words, R is antisymmetric if and only if:

$$\forall x \forall y ((x,y) \in R \land (y,x) \in R \to x = y)$$

> *Remark* Symmetric and antisymmetric are not opposite of each other. The following relation is both symmetric and antisymmetric:
>
> $$\{(a,a)\}$$
>
> Similarly, the following relation is neither symmetric nor antisymmetric:
>
> $$\{(a,b),(b,a),(a,c)\}$$

Definition: Transitive relations

A relation R on a set A **transitive** if and only if, whenever $(a,b) \in R$ and $(b,c) \in R$, then $(a,c) \in R$, for all $a, b, c \in A$.
In other words, R is transitive if and only if:

$$\forall x \forall y \forall z ((x,y) \in R \land (y,z) \in R \to (x,z) \in R)$$

Summary

We can draw the following table:

Name	Definition
Reflexive	$\forall x (x \in A \to (x,x) \in R)$
Symmetric	$\forall x \forall y ((x,y) \in R \to (y,x) \in R)$
Antisymmetric	$\forall x \forall y ((x,y) \in R \land (y,x) \in R \to x = y)$
Transitive	$\forall x \forall y \forall z ((x,y) \in R \land (y,z) \in R \to (x,z) \in R)$

Examples

Relation	Reflexive	Symmetric	Antisymmetric	Transitive
$\{(a,b) \vert a \leq b\}$	Yes	No	Yes	Yes
$\{(a,b) \vert a > b\}$	No	No	Yes	Yes
$\{(a,b) \vert a = b \lor a = -b\}$	Yes	Yes	No	Yes
$\{(a,b) \vert a = b\}$	Yes	Yes	Yes	Yes
$\{(a,b) \vert a = b + 1\}$	No	No	Yes	No
$\{(a,b) \vert a + b \leq 3\}$	No	Yes	No	No

Number of relations on a set

We wonder how many relations can be made on a set A. We know that:

$$|A \times A| = |A| \cdot |A| = |A|^2$$

Then, every subset of $A \times A$ can be a relation, so we are looking at the power set:

$$|\mathcal{P}(A \times A)| = 2^{|A \times A|} = 2^{|A|^2}$$

So, there are often many possible relations, but it is finite on a finite set.

Tuesday 19th October 2021 — **Lecture 9 : Equivalence relations and partial orderings**

7.3 Equivalence relations

Definition: Equivalence A relation on a set A is called an **equivalence relation** if it is reflexive, symmetric, and transitive.
Two elements a and b are related by an equivalence relation are called **equivalent**, often written $a \sim b$.

> *Motivation* Let's suppose we have coins of different denominations (4 times 1, 3 times 2 and 2 times 5, for example). Then we may want to make a relation R which says that "coin x has the same value as coin y", because they are different coins but to a shopper it would make no difference to pay with one two francs coin or another of the same value.
>
> We realise that R, in this cas, is symmetric, reflexive and transitive. We generalise equivalence relations using those properties.

Example 1 Let the following relation:

$$R_{minus} = \{(a, b) \in \mathbb{R} \times \mathbb{R} |\ a - b \in \mathbb{Z}\}$$

We can see that two numbers ($a = 3.142$ and $b = 2.142$, for example) are in the relation if and only if they have the same decimal digits.

1. We see that it is reflexive since

$$a - a = 0 \in \mathbb{Z}$$

2. It is symmetric since

$$(a-b) \in \mathbb{Z} \implies (b-a) = -\underbrace{(a-b)}_{\in \mathbb{Z}} \in \mathbb{Z}$$

3. It is transitive since

$$(a-b), (b-c) \in \mathbb{Z} \implies a - c = \underbrace{(a-b)}_{\in \mathbb{Z}} + \underbrace{(b-c)}_{\in \mathbb{Z}} \in \mathbb{Z}$$

So, it is an equivalence.

Example 2 Let the following relation:

$$R_{divides} = \{(a, b) \in \mathbb{N} |\ a \text{ divides } b\}$$

We notice that it is not an equivalence, since it is not symmetric. For example, $2 \mid 4$ but $4 \nmid 2$.

Definition: Equivalence class Let R be an equivalence relation on a set A. The set of all elements that are related to an element a of A is called the **equivalence class** of a, denoted $[a]_R$, or $[a]$.
In other words:

$$[a]_R = \{s \in A |\ (a, s) \in R\}$$

If $b \in [a]_R$, then b is called a **representative** of this equivalence class.

Example Let's come back to our previous equivalence relation:

$$R_{minus} = \{(a, b) \in \mathbb{R} \times \mathbb{R} |\ a - b \in \mathbb{Z}\}$$

We have the following equivalence classes:

$$[0]_{\mathcal{R}_{\text{minus}}} = \mathbb{Z}$$

$$[3.142]_{\mathcal{R}_{\text{minus}}} = \{3.142, 4.142, 2.142, \ldots\}$$

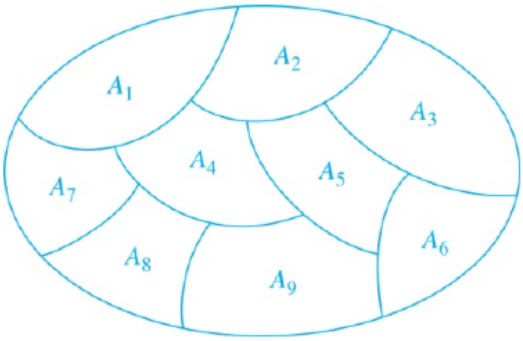
Theorem Let R be an equivalence relation on a set A. These statements for elements a and b of A are equivalent:

1. $R(a, b)$
2. $[a] = [b]$
3. $[a] \cap [b] \neq \emptyset$

Definition: Partition of a set A **partition** of a set is a collection of disjoint nonempty subsets of S that have S as their union. More formally, the collection of subsets $A_i \subset S$ form a partiition of S if and only if:

$$A_i \neq \emptyset, \quad i \neq j \implies A_i \cap A_j = \emptyset, \quad \bigcup_i A_i = S$$

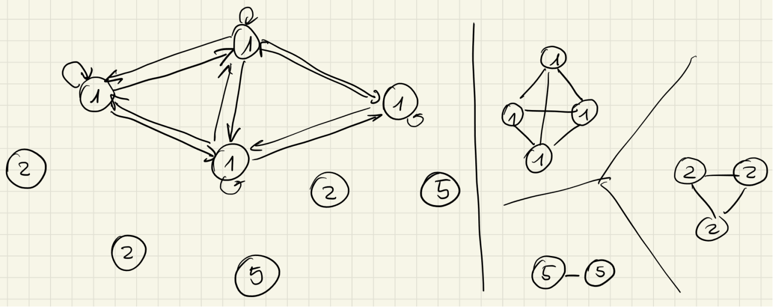
Illustration



Theorem Let R be an equivalence relation on a set S.
Then, the equivalence classes of R form a partition of S. Conversely, given a partition $\{A_i\}$ of the set S, there is an equivalence relation R that has the sets A_i as its equivalence classes.

Other formulation Set partitions and equivalence classes are equivalent: we can construct one from another.

Illustration Let's take back the example we did to introduce this subject, i.e. two coins are in an equivalence relation if and only if they have the same denomination. We can make the following drawings:



On the left, we see the equivalence relation, on the right the set partition.

7.4 Partial orderings

Definition A relation R on a set S is called a **partial ordering**, or **partial order**, if it is reflexive, antisymmetric, and transitive
A set together with a partial ordering R is called a **partially ordered set**, or **poset**, and is denoted by (S, R).

> *Motivation* Let's say we still have our coins, but we want to sort them. Let's pick R to be the relation "coin x has at least the value of coin y".
>
> We can see that $\mathcal{R}$ is transitive, reflexive and anti-symmetric. This is a **total order**.
>
> Let's say we want to compare purses. Let's pick $\mathcal{R}$ be the relation "purse x has at least as many coins of any value as purse y". If we have (a, b), a purse of a two francs coins and b one francs coin, then we have $R((2,1),(1,1))$ and $\mathcal{R}((\infty, \in),(\infty,\infty))$. However, there is no relation between $(1, 2)$ and $(2, 1)$, we cannot compare them. This is called a **partial ordering**.

Definition: Comparability The elements a and b of a poset $(S, \preccurlyeq)$ are **comparable** if either $a \preccurlyeq b$ or $a \preccurlyeq b$. When a and b are elements of S so that neither $a \preccurlyeq b$ nor $a \preccurlyeq b$, then a and b are called **incomparable**.

> *Remark* Note that the symbol $\preccurlyeq$ is often used to denote the relation in posets. This is done as an analogy to $\leq$, since $(S, \leq)$ is a poset for any set S.

Example 1 We can see that $(\mathbb{Z}, \geq)$ is a poset.
Indeed, it is easy to show that it is reflexive, since $a \geq a$ is true for all integers. It is clear that $a \geq b$ and $b \geq a$ implies $a = b$, so it is anti-symmetric. Finally, it is transitive, since $a \geq b$ and $b \geq c$ implies that $a \geq c$.

Example 2 We can see that $(\mathbb{Z}_+, |)$ is a poset:

- **Reflexive:** $a \mid a$ is true for all integers.
- **Anti-symmetric:** Let's say we have $a \mid b$ and $b \mid a$. In other words, we have $a = k_1 b$ and $b = k_2 a$. Thus:

$$a = k_1 b = k_1 k_2 a \implies k_1 k_2 = 1$$

  Since $k_1, k_2 \in \mathbb{Z}$, we have $k_1 = k_2 = 1$, and therefore $a = b$.
- **Transitive:** Let's say we have $c \mid b$ and $b \mid a$, i.e. $c = k_1 b$ and $b = k_2 a$. So:

$$c = k_1 b = k_1 k_2 a \implies c \mid a$$

Example 3 We can see that $(\mathcal{P}(S), \subseteq)$ is a poset:

- **Reflexive:** $A \subseteq A$, indeed.
- **Anti-symmetric:** We know by definition of set equality that:

$$A \subseteq B, B \subseteq A \implies A = B$$

- **Transitive:** We know that:

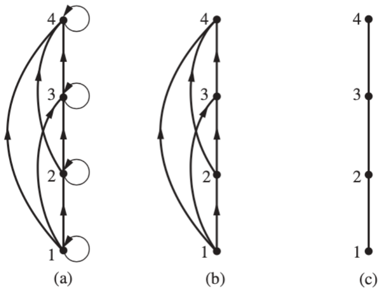
$$A \subseteq B, B \subseteq C \implies A \subseteq C$$

Example 4 The following are not posets, since they are not reflexive:

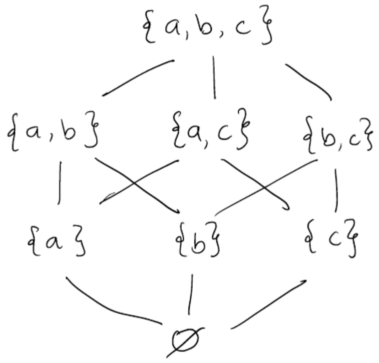
$$(\mathbb{Z}, \neq), \quad (\mathbb{Z}, \nmid), \quad (\mathbb{R}, <)$$

Hasse diagrams We know that all relations are reflexive, transitive and anti-symmetric.
We can draw a directed graph (a). Then, we can omit self-loops (since they are all reflexive) (b). Finally, we can remove transitive edges, and assume that arrows point

upwards (no arrow point in both direction since relations are anti-symmetric). The last diagram is a Hasse diagram.



Example Let's draw the Hasse diagram of $(\mathcal{P}(\{a, b, c\}), \subseteq)$:



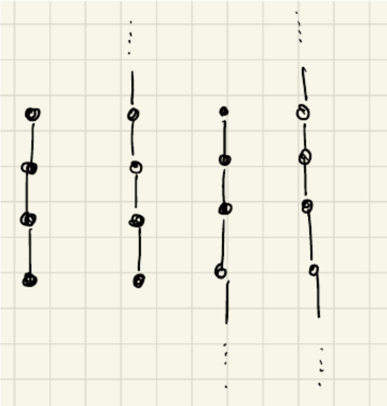
Personal note Fun fact, this draws exactly a 3D cube.

Definition If $(S, \preccurlyeq)$ is a poset and every two elements of S are comparable, then S is called a **totally ordered** or **linearly ordered set**, and $\preccurlyeq$ is called a **total order** or a **linear order**.
We say that $(S, \preccurlyeq)$ is a **well-ordered** set if it is a poset such that $\preccurlyeq$ is a total ordering, and every nonempty subset of S has a least element.

Note One of the defining axiom of the natural numbers, is its well-ordering, i.e that $(\mathbb{N}, \leq)$ is well-ordered.

Example $(\mathbb{Z}, \leq)$ is totally ordered, and $(\mathbb{Z}_+, |)$ and $(\mathcal{P}(S), \subseteq)$ (with $|S| > 1$) are not totally ordered.

Hasse Diagrams We can draw the following four Hasse diagrams.



They are all Hasse diagrams of totally ordered sets, and the two first ones are well ordered.

Definition Let $(S, \preccurlyeq)$ be a partially ordered set.
An **upper bound** $u \in S$ of $A \subseteq S$ is an element such that:

$$a \preccurlyeq u, \quad \forall a \in A$$

A **lower bound** $u \in S$ of $A \subseteq S$ is an element such that:

$$u \preccurlyeq a, \quad \forall a \in A$$

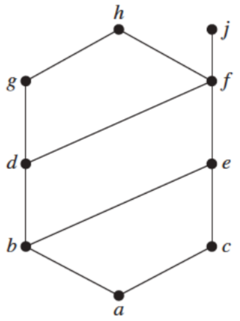
> *Remark* Note that u is an element of S, thus it is not necessarily element of A.
>
> For example, using $(\mathcal{P}(\{a, b, c\}), \subseteq)$ (its Hasse diagram can be found hereinabove), if we have $A = \{\{a\}, \{b\}, \{c\}\}$, then the only possible upper bound is $\{a, b, c\}$.

A **least upper bound** $u \in S$ of $A \subseteq S$ is an upper bound of A such that it is less than every other upper bound of A.
A **greatest lower bound** $u \in S$ of $A \subseteq S$ is a lower bound of A such that it is greater than every other bound of A.

> *Remark* The least upper bound and greatest lower bound of a subset A are unique, if they exist (this follows directly from anti-symmetry). Note that they do not always exist (if we have bounds that are not comparables, for instance).

Wednesday 20th October 2021 — **Lecture 10 : Sequences and countable infinities**

Example Let's draw the following Hasse diagram:



Then, we can see that h is an upper bound for $\{a, e, d\}$; f is a least upper bound for $\{a, e, d\}$ and $\{j, h\}$ has no upper bound.

Definition: Lattices A partially ordered set in which every pair of elements has booth a least upper bound and a greatest lower bound is called a lattice.

> *Example* $(\mathcal{P}(S), \subseteq)$ is a lattice. Indeed, the least upper bound of two subsets is $A \cup B$ and the greatest lower bound is $A \cap B$.

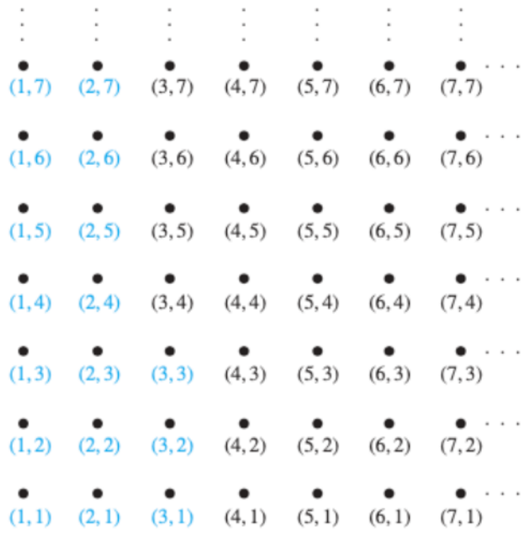
Definition: Lexicographic ordering Given two posets $(A_1, \preccurlyeq_1)$ and $(A_2, \preccurlyeq_2)$, the **lexicographic ordering** on $A_1 \times A_2$ is defined by saying that $(a_1, a_2) \preccurlyeq (b_1, b_2)$ if:

$$a_1 \preccurlyeq b_1 \vee (a_1 = b_1 \wedge a_2 \preccurlyeq_2 b_2)$$

This definition can easily be extended to a lexicographic ordering on a n-ary Cartesian product.

> *Motivation* This is basically the generalised idea of alphabetical order.

Example Let's consider $(\mathbb{Z} \times \mathbb{Z}, \leq)$. All ordered pairs less than $(3, 4)$ are:



7.5 Sequences

Definition: Sequences A **sequence** is a function from a subset of the integers — usually $\mathbb{Z}_+$ or $\mathbb{N}$ — to a set S.
Let $f : \mathbb{Z}_+ \mapsto S$ be the function that defines a sequence. We write a_n to denote the image $f(n)$ of integer n. We call a_n a **term** of the sequence.

Intuition Sequences are basically ordered lists of elements of a set.

Example Let (a_n) denote the sequence that is defined by $a_n = \frac{1}{n}, n \geq 1$. Then:

$$a_1 = 1, \quad a_2 = \frac{1}{2}, \quad a_3 = \frac{1}{3}, \quad , \ldots$$

Types of sequences We can give name to the following sequences:

- **Arithmetic Progression:** $a_n = a + n \cdot d$, where a and d are given constants. For example, a car going at constant speed.
- **Geometric Progression:** $a_n = ar^n$. For example, interest rates.
- **Recurrence Relations:** $a_n = g(a_{n-1}, a_{n-2}, \ldots, a_{n-k})$ with k initial conditions.

Remark Note that the arithmetic progression can be defined recursively as $a_n = a_{n-1} + d$ and $a_0 = a$. Similarly, the geometric progression can be defined recursively as $a_n = ra_{n-1}$ and $a_0 = a$.

Sum We define

$$s_n = \sum_{j=1}^{n} a_j = a_1 + \ldots + a_n$$

Important sums If we have $a_j = d$, then:

$$s_n = \sum_{j=1}^{n} d = nd$$

Else, if we have $a_j = j$, then (Gauss is too strong, he found it when was in Elementary School when his teacher asked his class to sum numbers from 1 to 100, in order to have some peace):

$$s_n = \sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

Product

Concerning products, we define:

$$p_n = \prod_{j=1}^{n} a_j = a_1 a_2 \cdots a_n$$

Important products

If we have $a_j = r$, then:

$$p_n = \prod_{j=1}^{n} r = r^n$$

If $a_j = j$, then we have:

$$p_n = \prod_{j=1}^{n} j = n(n-1)\ldots 1 = n!$$

Telescoping series

Given $a_0, \ldots, a_n$, then:

$$\sum_{j=0}^{n}(a_j - a_{j-1}) = a_n - a_0$$

Proof

$$\begin{aligned}\sum_{j=1}^{n}(a_j - a_{j-1}) &= \sum_{j=1}^{n} a_j - \sum_{j=1}^{n} a_{j-1} \\ &= \sum_{j=1}^{n-1} a_j + a_n - \sum_{j=0}^{n-1} a_j \\ &= \sum_{j=1}^{n-1} a_j + a_n - \sum_{j=1}^{n-1} a_j - a_0 \\ &= a_n - a_0\end{aligned}$$

Example

We see that, if we pick $a_j = j^2$, then:

$$a_j - a_{j-1} = j^2 - (j-1)^2 = 2j - 1$$

So:

$$\sum_{j=0}^{n}(2j-1) = n^2 - 0^2 = n^2$$

Definition: String

A **string** is a finite sequence of characters from a finite set A (an alphabet).

Example

The **empty string** is represented by λ. The string *abcde* has length 5.

Lexicographic ordering on strings

A lexicographic ordering of strings can be defined using the ordering letters in the alphabet. This is the same ordering as that used in dictionaries.
We can note that strings with lexicographic ordering are well-ordered sets.

7.6 Countable sets

Cardinality comparison The cardinality of a set A is equal to the cardinality of a set B, denoted by $|A| = |B|$ if and only if there is a bijection from A to B (we could draw a line going from each element of a to each element of b, exactly one).
If there is an injection of from A to b, the cardinality of A is less than or equal to the cardinality of B, denoted $|A| \leq |B|$.
When $|A| \leq |B|$ and A and B have different cardinality, we say that the cardinality of A is less than the cardinality of B, denoted $|A| < |B|$.

Definition A set that is either finite or has the same cardinality as the set of positive integers, $\mathbb{Z}_+$, is called **countable**. Else, it is **uncountable**.
When a set is finitely countable, then its cardinality is its number of of elements. If it is countably infinite, its cardinality is $\aleph_0$.

Remark Note that $\aleph$ is read "aleph"; it is the first letter of the Hebrew alphabet.

Example 1 Let's show that the set of positive integers E is a countable set.
Indeed, let $f : \mathbb{Z}_+ \mapsto E$, $f(x) = 2x$. This want to show that f is a bijection.

Injective Suppose that $f(n) = f(m)$. Then:

$$2n = 2m \implies n = m$$

Therefore, f is injective.

Surjective Let t be an arbitrary even positive integer. We know that $t = 2k$ or some positive integer k. Thus, $f(k) = t$, so f is surjective.

Since f is bijective, we deduce that the E is a countable set.

Generalisation More generally, an infinite subset of a countable subset is countable.

Example 2: cardinality of power sets Let S be a set.
Then there exists no surjective function $f : S \mapsto \mathcal{P}(S)$. In other words, $|S| < |\mathcal{P}(S)|$.

Proof Let's assume for contradiction that such an f exists. Let's consider the following set:

$$T = \{s \in S | s \notin f(s)\}$$

Since $T \subseteq S$, we know that $T \in \mathcal{P}(S)$, and thus, since f is surjective, there exists $s_0 \in S$ such that $f(s_0) = T$.
If $s_0 \in T$, then $s_0 \notin f(s_0) = T$, which is a contradiction.
If $s_0 \notin T = f(s_0)$, then $s_0 \in T$ by definition of T, which is also a contradiction.

□

Theorem: Showing countability An infinite set S is countable if and only if it is possible to list the elements of the set in a sequence indexed by positive integers.

Poof $\implies$ Since S is countable, there exists a bijection $f : \mathbb{Z}_+ \mapsto T$. Therefore, we can form the sequence $a_1, \ldots, a_n, \ldots$ where:

$$a_1 = f(1), \ldots, a_n = f(n), \quad , \ldots$$

Proof $\Longleftarrow$ Since we can list the set in a sequence (a_n) indexed by the positive integers, then we can define the function $f(n) = a_n$, which is a bijection.

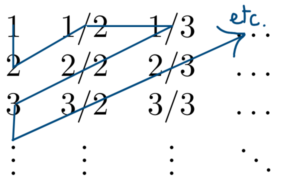
Example Let's show that the set of integers $\mathbb{Z}$ is countable. We can list its elements in a sequence:

$$0, 1, -1, 2, -2, 3, -3,$$

Thus, $\mathbb{Z}$ is countable.

Hilbert's Grand Hotel Hilbert's Grand Hotel has an infinite number of rooms, each occupied by a guest. However, we can always accommodate a new guest at this hotel.
Indeed, since the Grand Hotel is countable, we can list rooms: Room 1, Room 2, and so on. When a new guest arrives, we move the guest in Room n to Room $n+1$ for all positive integers n. This frees up Room 1, which we assign to the new guest, and all the current guests still have rooms.

Cardinality of rational numbers We want to show that rational numbers, $\mathbb{Q}$, are countable.
We can arrange them in a table, as $\frac{x}{y}$ in the x^{th} row and y^{th} column. We can follow a path that goes through all of them, and skip the ones we get multiple times ($\frac{1}{2}$ and $\frac{2}{4}$, for example).



$$\begin{array}{cccc} 1 & 1/2 & 1/3 & \cdots \\ 2 & 2/2 & 2/3 & \cdots \\ 3 & 3/2 & 3/3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

Theorem The union of a countable number of countable sets is countable.

Proof We can again put them in a table and follow a path that goes through all of them.

Set of finite strings The set of finite strings s over a finite alphabet A is countably infiniite.
Indeed, we can show that by listing all the strings. First, we list all the strings of length 0 in lexicographic order. Then, we list all the string of length 1 in lexicographic order. And we continue similarly.

Theorem: Real numbers The set of real numbers $\mathbb{R}$ is uncountable.

Proof Let us take $[0, 1] \subseteq R$. Let's assume for contradiction that it is countable. Thus, we have a first irrational number that can be written as

$$r_1 = 0.d_{11}d_{12}d_{13}d_{14}\ldots$$

Using its decimal expansion. We can do the same for $r_2, \ldots, r_i$. So, generally,

$$r_i = 0.d_{i1}d_{i2}\ldots d_{ii}\ldots$$

Let us now take the number

$$r = 0.d_1d_2d_3\ldots$$

where d_i is defined as followed:

$$d_i = \begin{cases} 3 \text{ if } d_{ii} \neq 3 \\ 4 \text{ if } d_{ii} = 3 \end{cases}$$

If r were in the list, then $r = r_i$. However, it would then mean that if $d_{ii} \neq 3$ then $d_i = 3$, and if $d_{ii} = 3$, then $d_i \neq 3$, which is a contradiction. Thus, r is not in the list and we have our contradiction.

□

Tuesday 26th October 2021 — **Lecture 11 : Introduction to algorithms**

Revision to countability The set $[0, 1]$ of real numbers is uncountable.

Proof We will use binary representation for more fun (for example: $(0.1)_2 = 0.5$ and $(0.11)_2 = 0.75$).

Let's suppose for contradiction that this set is countable. Thus, it can be written as a sequence (it does not matter how we write it, it must always be possible):

$$\begin{aligned} r_0 &= 0.1010101\ldots \\ r_1 &= 0.0010010\ldots \\ r_2 &= 0.1100110\ldots \\ &\vdots \\ r_i &= 0.d_{i1}d_{i2}\ldots d_{ii}\ldots \\ &\vdots \end{aligned}$$

Now we can construct $r = 0.011\ldots$ with every bit on the diagonal (in red), which we invert ($\overline{d_i} = 1 - d_{ii}$). Then it cannot be in the list, since the bit on the diagonal part would not be equal (we would have inverted it). Since it is not in the list, it is a contradiction to the fact that we can construct such a list.

□

Remark We can use the same argument to show that the power set of the natural numbers is uncountable. We can use a number as each bit saying "element i is in the power set".

Chapter 8

Algorithms

8.1 Introduction

Definition

An algorithm is a finite set of well-defined instructions to perform a specified task. Meaning, to perform a computation, to solve a problem, to reach a certain destination, and so on.
Note that we can do algorithms without computers.

History

There are algorithms ranging from 2500 BC in Babylon, that allowed to perform a division, Greece in 300 BC with finding prime number, from Muhammed ibn Musa al-Khwarizmi (the best, his name gave “algorithm”) in 780 BC with solving quadratic equations. Then, in the twentieth century, many people such as Hilbert, Gödel, Church, and Kleene worked on predicate logic and, Turing came up with the Turing Machine, which is the basis for algorithms on computers.

Example

Let’s suppose we want to find the maximum value in a finite sequence of integers. Then we can use the following algorithm:

1. Set the temporary maximum equal to the first integer in the sequence.
2. Compare the next integer in the sequence to the temporary maximum. If it is larger than the temporary maximum, then we set the latter to this integer.
3. When we get at the end of the list, our temporary maximum is the global maximum.

Specifying algorithms

We have multiple ways to describe algorithms: natural language, pseudo-code and programming languages.
Pseudo-code is an interemediate step, and it is used very often because it is more precise than natural language, but more general than a specfifc programming language. The idea is that if we know any typical programming language, then the syntax is clear enough to understand. For example:

```
procedure max(a: array of n integers):
    tmp_max := a[1]
    for i := 2 to n:
        if tmp_max < a[i] then tmp_max := a[i]
    return tmp_max
```

Typical problems

- **Searching problems:** Find an element in a list.
- **Sorting problems:** Put elements in a list in order.
- **Optimisation problems:** Determine the optimal value of a particular quantity over all possible inputs.

The two first ones are really important for managing data.

8.2 Searching algorithms

Searching problem Given a list $S = a_1, \dots, a_n$ of distinct elements, and some x. If $x \in S$, then it must return i such that $a_i = x$, else it must return 0.

Linear search algorithm The most simple one is to traverse the list, considering elements one by one, starting at the beginning. If we reach the element, then we have found it, else if we get out of the list without finding it, then it is not in it. Here is the pseudocode:

```
procedure linear_search(x: integer, a: array of n distinct integers):
    i := 1
    while(i <= n and x != a[i])
        i := i + 1
    if i == n + 1 then location := 0 else location := i
    return location
```

Example Let's say we are looking for $x = 2$:

sequence	3	5	1	7	2	1
$x \neq a_i$	T	T	T	T	F	
location	2	3	4	5		

Binary Search Let's assume that, this time, the input is a list of items in increasing order (they have been sorted previously).

1. We begin by comparing the element to be found with the middle element.
   (a) If the middle element is lower, the search proceeds with the upper half of the list.
   (b) If it is not lower, the search with the lower part of the list (including the middle position).
2. Repeat this process until we have a list of size 1.
   (a) If the element we are looking for is equal to the element in the list, the position is returned.
   (b) Otherwise, 0 is returned to indicate that the element was not found.

```
procedure binary_search(x: integer, a: array of n increasing integers)
    i := 1  // left-most index
    j := n  // right-most index
    while i < j
        m := floor((i + j)/2)
        if x > a[m] then i := m+1 else j := m
    if x = a[i] then location := i else location := 0
    return location
```

Example Let's say we are looking for 19 in the following list of 16 elements:

$$1, 2, 3, 5, 6, 7, 8, 10, \underline{12, 13, 15, 16, 18, 19, 20, 22}$$

Cutting in halves:

$$12, 13, 15, 16, \underline{18, 19, 20, 22}$$

Again:

$$18, 19, \underline{20, 22}$$

$$18, \underline{19}$$

$$19$$

We have a list of only one element, and it contains 19. So, we have found it.

8.3 Sorting algorithms

Sorting problems Given a list $S = a_1, \ldots, a_n$, we want to return a list where the elements are put in increasing order.

Bubble sort We will do different passes through the list:

1. In one pass, every pair of elements that are found to be out of order are interchanged.
2. Since the last element is guaranteed to be the largest after the first pass, in the second pass it does not need anymore to be inspected.
3. We can repeat this process, and every pass we have one less element to inspect.

```
procedure bubble_sort(a : array of n >= 2 real numbers):
    for i := 1 to n - 1
        for j := 1 to n-1
            if a[j] > a[j+1] then interchange(a[j], a[j+1])
```

Example Let's say we have the list

$$3, 2, 4, 1, 5$$

Then, comparing the first two elements, we realise that $3 < 2$ is wrong, so:

$$2, 3, 4, 1, 5$$

Now $3 < 4$ is good. However, $4 < 1$ is not, so we end up with:

$$2, 3, 1, 4, 5$$

We can finish our first pass since $4 < 5$. We indeed have that the greatest element, 5, is at the end of the list. We can repeat this process on the sub-list $2, 3, 1, 4, 5$ and we will end up with :

$$1, 2, 3, 4, 5$$

as expected.

Animations of this algorithms are really the best way to understand it.

3	2	2	2	2	2	2	2	1	1	1
2	3	3	3	3	3	1	1	2	2	2
4	4	4	1	1	1	3	3	3	3	3
1	1	1	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5
Pass 1				Pass 2			Pass 3		Pass 4	End

Insertion sort

1. Compare the second element with the first and puts it before the first if it is not larger.
2. Then the third element is put in corret position among the three first using linear search.
3. We continue this way every pass, increasing the size of our sorted subarray.

```
procedure insertion_sort(a: array of n >= 2 real numbers):
    for j := 2 to n
        i := 1
        while a[j] > a[i] and i < j  // move a[j] to the right
            position
            i := i + 1
        m := a[j]
        for k := 0 to j - i - 1  // shift elements to make space for m
             = a[j]
            a[j-k] := a[j-k-1]
        a[i] := m
```

Example

3	2	2	1	1
2	3	3	2	2
4	4	4	3	3
1	1	1	4	4
5	5	5	5	5

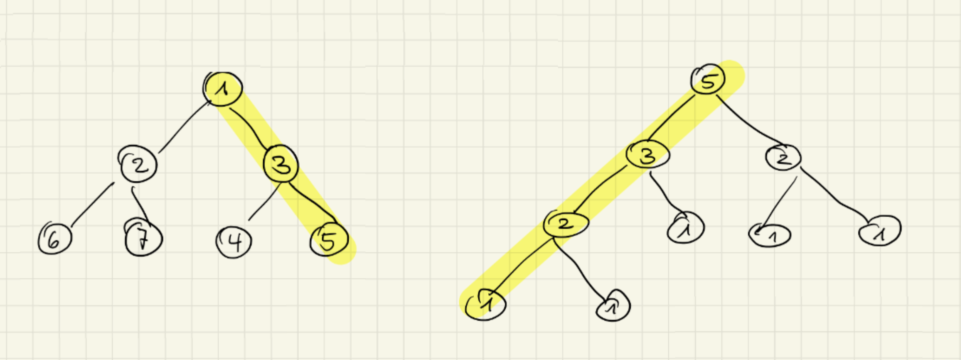
Wednesday 27th October 2021 — **Lecture 12 : Greed, coins and same-sex marriages**

8.4 Optimisation algorithms

Goal The goal is to minimise or maximise some parameter over all possible inputs. For example, if we want to find a route between two cities, determine how to encode messages with the fewest bits, and so on.

Greedy algorithms They are very short-sighted, and the only take the "best" choice at every step. This does not always gives us the optimal result. Sometimes, we have to go down a hill to get up a mountain. However, in many instances it does. Proving that the greedy algorithm is correct is required to use it.

Example



Firstly, the task is to find the longest path to get to the bottom of the left tree. A greedy algorithm would always go to the node with the greatest node, and it would work perfectly.
Now, let's say that the task to find the leaf with the largest value in the right tree (for example if it is a minimax algorithm for chess, we give a value to each position, and we want to get to the best in the long run). We will not go to the right leaf. Maybe sacrificing a knight allows us to get a rook.

Cashier's algorithm Let's say we have coins of different values. Coins can be for example "quarters" (25 cents), "dimes" (10 cents), "nickels" (5 cents) and "pennies" (1 cent). We want to be able to give a certain value using the least possible number of coins.
At each step, we choose the coin with the highest possible value that does not exceed the amount left.

```
procedure change(c : array of n > 0 values of coins, where c[1] > c[2]
     > ... > c[n])
    for i := 1 to r
        d[i] := 0  // number of coins of value c[i]
        while n >= c[i]
            d[i] := d[i] + 1
            n := n - c[i]  // modify amount left to give
    return d
```

Example Let's say we want to change for $n = 67$ cents. The highest coin available is 25, so we have 42 left. We can give a second quarter, and we have 17 cents left to give. Then we can give a dime, a nickel, a penny and another penny.

Proof of optimality We want to show that the change making algorithm for US coins is optimal.

Lemma The change of $n > 0$ cents using the least possible number of coins amongst quarters, dimes, nickels, and pennies

1. has at most 2 dimes, 1 nickel and 4 pennies;
2. cannot have 2 dimes and 1 nickel;
3. the total amount of change in dimes, nickels, and pennies cannot exceed 24 cents.

Proof of part (1) Let's suppose for contradiction that we have 3 dimes = 30 cents. Then we can replace them by 25 + 5 cents, which represent two coins.

2 nickels = 10 cents can be replaced by one coin of 10 cents, and 5 pennies = 5 cents can be replaced by one coin of 5.

Proof of part (2) We cannot have 2 dimes and 1 nickel, since we could replace them by one coin of 25.

Proof of part (3) Dimes nickels and pennies in total have less than 25, since else we could replace them by using a coin of 25.

Another way of seeing this is to see that we have at most 2 dimes, 1 nickel and 4 pennies, but we cannot have 2 dimes and 1 nickel, so we thus need to get rid of the penny. In other words, we have 2 dimes and 4 nickels, which represent 24 cents.

Theorem (optimality) The greedy change-making algorithm for US coins produces change using the fewest coins possible.

Proof Let c_{25}, c_{10}, c_5, c_1 the number of coins of the "optimal" solution and $\overline{c}_{25}, \overline{c}_{10}, \overline{c}_5, \overline{c}_1$ be the number of coins the algorithm gives.

Since the "optimal" solution is optimal, we know $\overline{c}_{25} \geq c_{25}$. Let's assume for contradiction that our program did not produce an optimal solution, meaning $\overline{c}_{25} > c_{25}$. It would then mean that the optimal solution would reach the same value, n, by using one less 25 cents coin. Thus, it needs to compensate this difference with the other coins:

$$10c_{10} + 5c_5 + c_1 > 24$$

This is a contradiction by the above lemma.

The same way, assuming that $\overline{c}_{10} > c_{10}$, then we would have

$$5c_5 + c_1 > 9$$

which is also a contradiction.

Again, assuming that $\overline{c}_5 \geq c_5$, then we would have $c_1 > 5$ which is another contradiction. We thus also conclude that $\overline{c}_1 = c_1$.

Available coins Optimality depends on the denominations available. If we have "quarters" (25 cents), "dimes" (10 cents) and "pennies" (1 cent), then the algorithm would not produce the minimum number of coins. For example, if we want to reach 31, it will use one quarter and 6 pennies, whereas we can reach it in 3 dimes and 1 penny.

Europe denominations In Europe we have $1, 5, 10, 20, 50$.

Lemma The solution with the smallest coins

1. has at most 4×1, 1×5, 1×10, 2×20;
2. cannot have 2×20 and 1×10 at the same time;

3. the total amount using $1, 5, 10, 20, 50$ cannot exceed 49.

We can then use this lemma to prove this denomination's optimality, using a proof similar to the one hereinabove.

Dropping coins

For denominations, $c_1, \dots c_n$ where $c_i | c_{i+1}$. The greedy algorithm is optimal (it is a sufficient condition, but not necessary).

Example for Europe

We see that it is not a necessary condition, since 20 does not divide 50. However, if we drop the 20, then we know that it definitely works for $1, 5, 10, 50$. However, for $1, 20, 50$ the greedy algorithm would not work (for example, when doing the change for 69 (nice) coins).

8.5 Stable matching

Goal

The task is to pair elements from equally sized two groups considering their preferences for members of the other group so that there are no ways to improve the preferences.

Definition of matching

Given a finite set A, a **matching** of A is a set of (unordered) pairs of distinct elements of A where any elements occurs in at most one pair (such pairs are called independent)

A **maximum matching** is a matching that contains the largest possible number of independent pairs.

Example

Let $A = \{1, 2, 3, 4\}$

- $\{1, 2\}$ and $\{(1, 3), (2, 4)\}$ are matchings.
- $\{2, 2\}$ and $\{(1, 2), (2, 4)\}$ are not matchings.
- $\{(1, 3), (2, 4)\}$ is a maximum matching.

Let $B = \{1, 2, 3, 4, 5\}$

- $\{(1, 3), (2, 4)\}$ and $\{(5, 3), (2, 4)\}$ are two different maximum matchings.

Definition of preferences

A **preference list** L_x defines for every element $x \in A$ the order in which the element prefers to be paired with. $x \in A$ prefers y to z if y precedes z on L_x.

Example

For example, if we have $A = \{\text{Lou, Glenn, Bobbie, Tyler}\}$. Then, we could have $L_{\text{Tyler}} = \{\text{Bobbie, Lou, Glenn}\}$.

Stability of matching

A matching is **unstable** if there are two pairs $(x, y), (v, w)$ in the matching such that x prefers v to y and v prefers x to w. In other words, each want to leave their partner.

A **stable** matching is a matching that is not unstable.

Example

For example, if we have $A = \{\text{Lou, Glenn, Bobbie, Tyler}\}$, and:

- $L_{\text{Lou}} = \{\text{Glenn, Bobbie, Tyler}\}$
- $L_{\text{Glenn}} = \{\text{Bobbie, Lou, Tyler}\}$
- $L_{\text{Bobbie}} = \{\text{Lou, Glenn, Tyler}\}$
- $L_{\text{Tyler}} = \{\text{Lou, Glenn, Bobbie}\}$

Let's begin with $\{(\text{Glenn, Lou}), (\text{Bobbie, Tyler})\}$. It is unstable since Glenn prefers Bobbie over Lou, and Bobbie prefers Glenn over Tyler.

Reordering our matching, we get $\{(\text{Glenn, Bobbie}), (\text{Lou, Tyler})\}$. It is unstable as well since Bobbie prefers to be with Lou, and Lou prefers to be with Bobbie.

Then, our new matching is $\{(\text{Lou, Bobbie}), (\text{Glenn, Tyler})\}$. Again, Lou prefers to be with Glenn, and Glenn prefers to be with Lou.

If we reorder another time our matchings, we will have come back to the initial matching, which shows that this will go on forever. It is possible that there does not exist any stable matching. (No one wants to stay with Tyler, poor them.)

Marriage problem

To guarantee the existence of a stable maximum matching, irrespective of the preference lists, it suffices to use a more stringent (harsher) pairing rule.

Given a set with even cardinality, we partition A into two disjoint subsets A_1 and A_2, where $A_1 \cup A_2 = A$ and $|A_1| = |A_2|$. A matching is a bijection from the elements of one set to the elements of the other set.

Put in simpler words, pairs can only consist of one element of A_1 and one of A_2 each. There is no way for an element from A_1 to be matched with another element of A_1.

The professor gave the example of boys and girls getting married, but let's use families instead. If we have two families A_1 and A_2, then members from one family cannot get married with a member of they own family (unless they come from northern France).

Existence of stable maximum matching

A greedy algorithm to construct a stable maximum matching for the marriage problem can be used to prove existence of stable matching. It is very efficient, and it gave a Nobel price to its creator. It is named the Gale-Shapley algorithm.

```
let M be the set of pairs under construction
Initially M = empty set
while |M| < |A1|:
    select an unpaired element x in A1
    let x propose to the first element y of A2 on Lx
    if y is unpaired then add the pair (x, y) to M
    else  // y is paired already
        let x' in A1 be the element that y is paired to
        if x' precedes x on Ly then remove y from Lx
        else  // x precedes x' on Ly
            replace (x', y) by (x, y) from M, and remove y from Lx'
```

Example

Let's say we have $A_1 = \{x_1, x_2, x_3, x_4\}$ and $A_2 = \{y_1, y_2, y_3, y_4\}$. Moreover, let

$$L_{x_i} = \{y_1, y_2, y_3, y_4\}$$

$$L_{y_1} = \{x_2, x_1, x_3, x_4\}$$

$$L_{y_2} = \{x_4, x_3, x_2, x_1\}$$

$$L_{y_3} = \{x_2, x_3, x_4, x_1\}$$

$$L_{y_4} = \{x_4, x_1, x_2, x_3\}$$

Let's pick x_1. Since it prefers y_1 over everyone else, and since y_1 is not paired yet, then we add (x_1, y_1) to M.

Now, picking x_2, it also prefers y_1. Since y_1 prefers x_2, then x_1 leaves y_1. M becomes (x_2, y_1).

We can pick x_1 again. y_1 is no longer on its list, so let's look at y_2. Since it is not paired yet, M becomes $(x_2, y_1), (x_1, y_2)$.

We can now pick x_3. y_1 is already paired and x_2 is before x_3 in L_{y_1}, then x_3 removes y from L_{x_3}. Considering y_2 now, since it prefers x_3 than x_1, x_1 leaves y_2, and M becomes $(x_2, y_1), (x_3, y_2)$.

Picking x_1 again. The first element on its list is y_3 and it is not paired yet, so M becomes $(x_2, y_1), (x_3, y_2), (x_1, y_3)$.

We can continue applying this reasoning over x_4.

At the end, the algorithm will give us the optimal solution.

Tuesday 2^{nd} November 2021 — **Lecture 13 : *O*-MG there's Alan**

8.6 The Halting Problem (yay Turing)

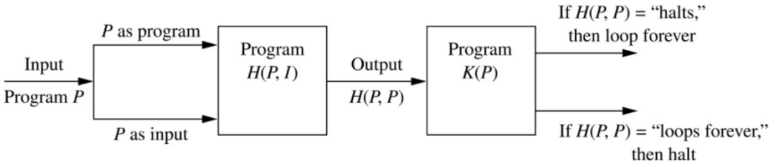
Question Can we solve any problem by an algorithm? Turing (☺) proved that nope!
It is similar to Gödel proving that not every theorem can be proven, with Gödel's incompleteness theorem.

Halting problem He defined an unsolvable problem, the **halting problem**: Can we develop a procedure that takes as input a computer program along with its input and determines whether the program will eventually halt when used with that input?
For example, a program with a `while(true)` loop would never stop.

Theorem The halting problem cannot be solved using an algorithm.

Proof We are not going to do a completely formal one, since we would else need to know exactly what is a procedure, the input and output and how a procedure can be encoded as a string.

Let's assume for contradiction that there is such a procedure, and let's call it $H(P, I)$, which says whether the program P with the inputs I halts. It could for example output "halts" or "loops forever" depending on what P does with the inputs I.
Let's construct a procedure $K(P)$. If $H(P, P)$ outputs "loops forever", them $K(P)$ halts. If $H(P, P)$ outputs "halt", then $K(P)$ goes into an infinite loop. This can be easily constructed, since it is basically an if clause.



Now, we can call K with K as input, i.e. $K(K)$. If the output of $H(K, K)$ is "loops forever", then $K(K)$ halts, which is a contradiction. If the output of $H(K, K)$ is "halts", then $K(K)$ loops forever, which is also a contradiction.

□

Other example Let the following code:

```
define function f(n) = n/2 if n is even or 3n + 1 if n is odd
procedure collatz(n):
    c := f(n)
    while c != 1
        c := f(c)
    return 0
```

If we have $n = 3$, then we will have the following sequence of c's:

$$10, 5, 16, 8, 4, 2, 1$$

Now, let's pick $n = 7$:

$$22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1$$

The Collatz conjecture ("conjecture de Syracuse") says that `collatz(n)` halts for all integers n. This is considered one of the most difficult problem nowadays.
Now we can try to write a program that halts when the Collatz conjecture is false. If it was possible, then it would be very easy to prove mathematical problems.

Personal note: I'm not sure I agree with this case, we "only" have to prove the Collatz conjecture is true or false, and then we have a one-line-long program returning `true` or `false` depending on what we proved. I think a better example is that, we know some theorems cannot be proven (maybe the Collatz conjecture is one of those, but we don't know) thanks to Gödel's incompleteness theorem; thus writing a program that predicts the result of such theorem is a contradiction to Gödel's theorem.

Chapter 9

Measuring complexity

Amount of details

We have different amounts of details we can take into account.

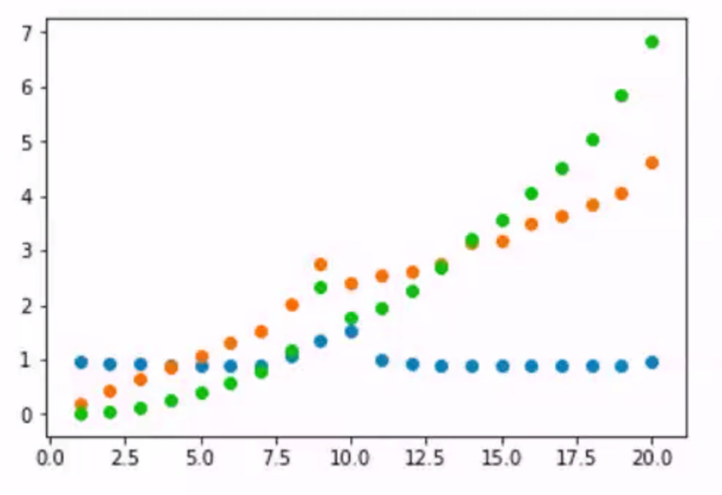
1. Precisely count everything involved (computer instructions, disk accesses, ...).
   It is very inconvenient and will not be the same on every computers.
2. Measure the amount of time a program takes on a computer.
   If sorting a list of n elements takes 2 s, then we would not be able to know how much time it takes to sort a list of $2n$ elements.

This last sentence is important, it would be great if, for example, knowing that a program sorted n elements in s seconds, we could deduce how much time it would take to sort m elements.

Example

Assume we want to run the following algorithm:

1. Create a list of 3000 random numbers and sort it using bubble sort.
2. Create n lists of length 1500 and sort them using bubble sort.
3. Create a list of length $400 \cdot n$ and sort it using bubble sort.



We can see some imperfections, it is due to the fact that it was done on a real computer, so there might be another task coming, and so.

The blue dots are constants, it is the first task. The orange dots are linear, they are the second task. And, the green dots are quadratic, they represent the third task.

We have approximately:

1. 1000 ms for the first task; it is independent of n
2. $200n$ ms for the second task
3. $1.5n^2$ ms for the third task

We can thus estimate the time spent using:

$$f(n) = 1.5n^2 + 200n + 1000$$

So, for a small n, then the first task dominates the cost. Then, some time later, it's the second one. However, finally, it's the third tasks that takes over. In other words, ultimately, only $g(n) = 1.5n^2$ is relevant, the others do not have any impact.

Observations In the last paragraph, we realised that, if we have the following function:

$$f(n) = g(n) + h(n) + \ldots + t(n)$$

Then, only the "ultimately largest" of $g, h, \ldots, t$ determines f's behaviours when n gets large.
Moreover, $f(n)$'s growth rate is independent of multiplicative constants in $g(n)$, since on different computers, there might be different CPUs, so, anyway, there would be a constant here (it might be twice as slow, for example). We can see that with another property:

$$\frac{g(m)}{g(n)} = \frac{cg(m)}{cg(n)}$$

Consequence So, when considering a runtime function for $f(n)$, we focus on parts that grows the fastest when $n \to \infty$, and we forget about multiplicative constants.
Using those two ideas, we come up with the Big-O notation.

Big-*O* notation Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there are constants C and k such that:

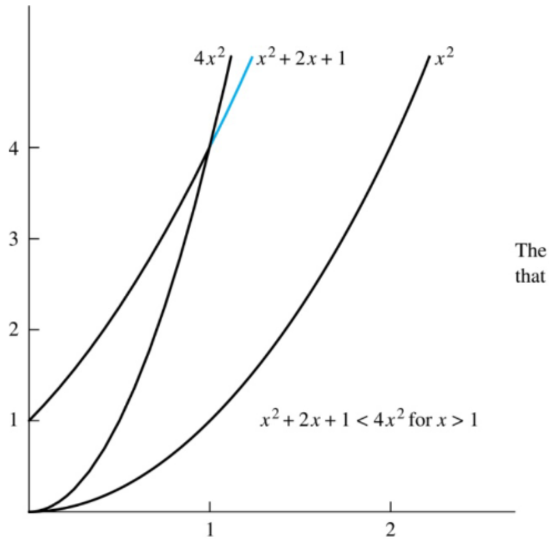
$$|f(x)| \leq C|g(x)|, \quad \forall x > k$$

This is read as "$f(x)$ is big-O of $g(x)$" or, "g asymptotically dominates f".
The constants C and k are called **witnesses** to the relationship "$f(x)$ is $O(g(x))$".
We only need to find one pair of witnesses to prove the relation.
We will write "$f(n)$ is $O(g(n))$" when we use natural numbers. The definition is the same.

> *Remark* Sometimes "$f(x) = O(g(x))$" is used in literature, however it's an abuse of the equality sign. We could use $f(x) \in O(g(x))$ since $O(g(x))$ is the set of functions that have this big-O notation, but nobody uses that.

Example We want to show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$.
An interesting fact is that, for $x > 0$, then $f(x) > x^2$. However, we will be able to prove the property using the multiplicative constant.

> *Proof* If $x > 1$, then $x^2 > x$ and $x^2 > 1$. So:
>
> $$|f(x)| = f(x) = x^2 + 2x + 1 < x^2 + 2x^2 + x^2 = 4x^2$$
>
> So, with $C = 4$ and $k = 1$, then:
>
> $$|f(x)| < Cx^2, \quad \forall x > k$$



The part of the graph of $f(x) = x^2 + 2x + 1$ that satisfies $f(x) < 4x^2$ is shown in blue.

Example 2 We want to show that x^2 is not $O(x)$.
Let's assume for contradiction that there exists k and C such that

$$x^2 \leq Cx, \forall x > k$$

Thus, we get that:

$$x^2 \leq Cx \implies x \leq C$$

But it is not true $\forall x > k$ if we have $x \leq C$ (the professor used another method, but I have to admit I prefer this one).

□

Examples

- 75 is $O(1)$ and 1 is $O(75)$
- 1 is $O(x)$ but x is not $O(1)$.
- x is $O\left(x^2\right)$ but x^2 is not $O(x)$.
- x^2 is $O\left(x^2\right)$
- x^2 is $O\left(x^3\right)$
- x^2 is $O\left(6x^2 + x + 3\right)$ and $6x^2 + x + 3$ is $O\left(x^2\right)$

Note that $O(75)$ and $O\left(6x^2 + x + 3\right)$ are unusual.

Theorem (Big-O for polynomials) Let $f(x) = a_n x^n + \ldots + a_1 x + a_0$, where $a_0, \ldots, a_n$ are real numbers and $a_n \neq 0$. Then, $f(x)$ is $O(x^n)$.

Proof If $x > 1$, then $x^n > x^{n-k}$ for $k = 1, \ldots, n$. So:

$$\begin{aligned}
\left|a_n x^n + a_{n-1} x^{n-1} \ldots + a_0\right| &\leq |a_n x^n| + \left|a_{n-1} x^{n-1}\right| \ldots + |a_0| \\
&\leq |a_n||x^n| + |a_{n-1}|\left|x^{n-1}\right| + \ldots a_0 \\
&\leq |a_n| x^n + |a_{n-1}| x^n + \ldots + |a_0| x^n \\
&= (|a_n| + |a_{n-1}| + \ldots + |a_0|) x^n \\
&= C x^n
\end{aligned}$$

Monomials Let n, m constants, with $n > m$. We have that x^m is $O(x^n)$ but x^n is not $O(x^m)$.

Logarithms Let a, b, n, m be constants, with $a > 0$, $b > 0$ and $n > m$.
$\log_b(x^m)$ is $O(\log_a(x^n))$ and $\log_a(x^n)$ is $O(\log_b(x^m))$. Indeed:

$$\log_b(x^m) = m \log_b(x) = \frac{m}{\log(b)} \log(x)$$

More specifically, and more importantly, they are all $O(\log(x))$. For a logarithm, its basis and the power inside do not matter.

Factorial function

Let the following function:

$$f(n) = n! = 1 \cdot 2 \cdot \ldots \cdot n$$

Then, we have:

$$n! = 1 \cdot 2 \cdot \ldots \cdot n \leq n \cdot n \cdot \ldots \cdot n = n^n$$

So, $n!$ is $O(n^n)$ taking $C = 1$ and $k = 1$.
Moreover, we see that

$$\log(n!) \leq \log(n^n) = n \log(n)$$

Hence, we deduce that $\log(n!)$ is $O(n \log(n))$ taking $C = 1$ and $k = 1$. This is a big-O that is often found, which grows a bit faster than linearly. Since it is a combination of a logarithmic and a linear function, we call it linearithmic. It's considered good for an algorithm.

Combinations of functions

- It is transitive. In other words, if $f(x)$ is $O(g(x))$ and $g(x)$ is $O(h(x))$ then $f(x)$ is $O(h(x))$.
- It preserves multiplication. In other words, if $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 \cdot f_2)(x)$ is $O((g_1 \cdot g_2)(x))$.
- We can add functions that have the same growth without changing anything. In other words, if $f_1(x)$ and $f_2(x)$ are both $O(g(x))$, then $(f_1 + f_2)(x)$ is also $O(g(x))$.
- More precisely, if we add two functions that have different growth, then only the one with the highest matters (just like black lives, they matter). In other words, if $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|))$.

Proof of the last fact

We know that there exists k_1, k_2, C_1, C_2 such that

$$|f_1(x)| \leq C_1 |g_1(x)|, \quad x \geq k_1$$

$$|f_2(x)| \leq C_2 |g_2(x)|, \quad x \geq k_2$$

Let's take $k = \max(k_1, k_2)$, and $g(x) = \max(g_1(x), g_2(x))$. Then, for $x > k$, we have:

$$\begin{aligned} |f_1(x) + f_2(x)| &\leq |f_1(x)| + |f_2(x)| \\ &\leq C_1 |g_1(x)| + C_2 |g_2(x)| \\ &\leq C_1 |g(x)| + C_2 |g(x)| \\ &= (C_1 + C_2)|g(x)| \end{aligned}$$

So, taking $C = C_1 + C_2$, this property holds as well.

Summary

Let's take functions $f : \mathbb{R}_+ \mapsto \mathbb{R}$. We have the following order of growth:

- **Constant:** $O(1)$
- **Logarithmic:** $O(\log x)$
- **Poly-logarithmic:** $O\Big((\log x)^d\Big)$, where $d > 1$
- **Linear:** $O(x)$
- **Linearithmic:** $O(x \log x)$
- **Polynomial:** $O(x^d)$, where $d > 1$
- **Exponential:** $O(b^x)$, where $b > 0$
- **Factorial:** $O(x^x)$

Until polynomial growth, it is still acceptable. However, exponential growth or more is very bad for an algorithm. Even though there is only one step in this list between polynomial and exponential growth, the difference is huge.

Wednesday 3rd November 2021 — **Lecture 14 : Big-Theta and algorithm complexity**

Power of logarithms

We want to show that $\log_2(x)^t$ is $O(x^\varepsilon)$, where $t, \varepsilon > 0$.
It is rather obvious that $x < 2^x$ when $x > 0$ (there are thousand of proofs possible, such as induction for natural numbers, or Taylor series). So:

$$\begin{aligned}
&x < 2^x \\
\implies &\log_2(x) < x \\
\implies &\log_2\left(x^{\frac{\varepsilon}{t}}\right) < x^{\frac{\varepsilon}{t}} \\
\implies &\frac{\varepsilon}{t}\log_2(x) < x^{\frac{\varepsilon}{t}} \\
\implies &\log_2(x) < \frac{t}{\varepsilon}x^{\frac{\varepsilon}{t}} \\
\implies &\log_2(x)^t < \left(\frac{t}{\varepsilon}\right)^t x^\varepsilon
\end{aligned}$$

Then, taking $k = 1$ and $C = \left(\frac{t}{\varepsilon}\right)^t$, we find that $\log_2(x)^t$ is $O(x^\varepsilon)$.
We have to be careful about the constants. If $t = 1000$ and $\varepsilon = 0.0001$, then $C = 10^6$. So, even though in the long term it would grow slower than x^ε, it is not the case for values that we would use commonly. In other words, even though the big-O of an algorithm is worse than one of another, does not necessarily means that it is better (personal note: see galactic algorithms; don't you want to do a 1729-dimensional (nice number) Fourier transform to multiply two numbers? It's more optimised in the long run though.).

Symmetry

We wonder if we know that f is not $O(g)$, then can we conclude that g is $O(f)$? The answer is no.
Let the following functions:

$$f(n) = \begin{cases} n! & \text{if } n \text{ is even} \\ (n-1)! & \text{if } n \text{ is odd} \end{cases}$$

$$g(n) = \begin{cases} (n-1)! & \text{if } n \text{ is even} \\ n! & \text{if } n \text{ is odd} \end{cases}$$

First, let's prove that f is not $O(g)$. For all C, for all $n > C$, if n is even:

$$f(n) = n! = n(n-1)! > C(n-1)! = Cg(n)$$

We can use the same way when n is odd to prove that g is not $O(f)$.

9.1 Big-Ω and Big-Θ

Big-Omega notation

Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Omega(g(x))$ if there are constants C and k with $C > 0$ such that

$$|f(x)| \geq C|g(x)| \quad x > k$$

Remark This is basically the inverse of the big-O notation. Big-O gives an upper bound of the growth of a function, while big-Omega gives a lower bound.

Terminology We say "$f(x)$ is big-Omega of $g(x)$".

> *Symmetric* $f(x)$ is $\Omega(g(x))$ if and only if $g(x)$ is $O(f(x))$.

Example Let $f(x) = 8x^3 + 5x^2 + 7$. It is $\Omega\big(x^3\big)$, since x^3 is $O(f(x))$.

Bit-Theta notation Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers.
The function $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$.

> *Terminology* We say that "f is big-Theta of $g(x)$", "$f(x)$ is of order $g(x)$", or "$f(x)$ and $g(x)$ are of the same order".

> *Symmetry* When $f(x)$ is $\Theta(g(x))$ then also $g(x)$ is $\Theta(f(x))$.
> It is equivalent to say that $f(x)$ is $O(g(x))$ and $g(x)$ is $O(f(x))$.

> *Literature* Sometimes people in literature use the big-O notation when they mean the big-Theta.

Example Since both $8x^3 + 5x^2 + 7$ is $O\big(x^3\big)$ and x^3 is $O\big(8x^3 + 5x^2 + 7\big)$, we thus know that $8x^3 + 5x^2 + 7$ is $\Theta\big(x^3\big)$.

Polynomials By the theorem we saw in the last lesson, we get that $f(x) = a_n x^n + \ldots + a_0$ is $\Theta(x^n)$.

Little-o We say that "$f(x)$ is $o(g(x))$" if:

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$$

Example We see that x^2 is $o\big(x^3\big)$, but $x^2 + x + 1$ is not $o\big(x^2\big)$.

Little-*o* and Big-*O* If $f(x)$ and $g(x)$ are functions such that $f(x)$ is $o(g(x))$, then $f(x)$ is $O(g(x))$.

> *Warning* The reciprocal is not true. For example, $x^2 + x + 1$ is $O\big(x^2\big)$, but not $o\big(x^2\big)$.

Linearithmic and linear functions We want to show that x is $O(x \log x)$. We see that x is $o(x \log x)$:

$$\lim_{x\to\infty} \frac{x}{x\log x} = \lim_{x\to\infty} \frac{1}{\log x} = 0$$

So, we deduce that x is $O(x \log x)$.

Polynomial and exponential functions We want to show that x^d is $O(b^x)$, where $b > 1$ and $d > 0$. We can use L'Hospital's rule to compute the following limit:

$$\lim_{x\to\infty} \frac{x^d}{b^x} = \lim_{x\to\infty} \frac{dx^{d-1}}{\ln(b)b^x} = \lim_{x\to\infty} \frac{d(d-1)x^{d-2}}{\ln(b)^2 b^x} = \ldots = \lim_{x\to\infty} \frac{d(d-1)\ldots 2\cdot 1}{\log(b)^d b^x} = 0$$

So, we can deduce that x^d is $o(b^x)$, therefore x^d is $O(b^x)$.

The gap between polynomial and exponential We know there is a tremendous gasp between the polynomial and the exponential growth, we are looking for a function in between.
Let's write $x = b^{\log_b(x)}$. Then, we have $x^d = b^{d\log_b(x)}$.
We notice that there is something between logarithms and linear growth, polylogarithms. So, we can choose $\log_b(x)^c$ for $c > 1$ as exponent of b:

$$b^{\log_b(x)^c}$$

This is called "quasi-polynomial" and it stands between polynomial and exponential growth.

> *Remark* the best algorithm to solve "graph isomorphism" is quasi-polynomial.

9.2 Computational complexity

Task Given an algorithm, we want to know how efficient it is for solving a problem given an input of a particular size.

Complexity There are two types of complexity. Indeed, we may want to know how much time an algorithm uses to solve the problem for an input of a given size (**time complexity**), or how much computer memory it uses to solve the problem for an input of a given size (**space complexity**).
It is important to understand the concept of complexity, in order to know whether it is possible to use an algorithm for inputs of a particular size, or to choose the best algorithm in a context, among all the ones that solve the task.

Time complexity We will focus on time complexity. If the algorithm is sequential, all operations are executed in order (other types of algorithms might for example be parallelised, so we will restrain ourselves to the sequential ones). In this case, we can say that time complexity is equivalent to the number of operations that are performed.
We can use big-O and big-Theta notations to describe the time complexity.
We will ignore implementation details (such as the data structures that are used, the hardware and the software) because it would only make everything more complicated, for no reason.

Determining time complexity We determine the number of basic operations, such as addition, multiplication, etc. We assume that all operations use a constant time (it is not exactly right, but since we only want to know how time grows when the input size grows, then it is no big problem).
We can ignore the small details, such as the "house-keeping" (initialisation of the first variables, etc.)

Worst-case time complexity We want to be pessimistic (if we want to sort a list and we get an already sorted list, then some sorting algorithms will be very fast; we choose exactly the inverse, the worst possible case). It gives us an upper bound on the number of operations an algorithm uses to solve a problem with input of a particular size.
We can also study average case time complexity of an algorithm, the average number of operations, however it is much harder to do. We will thus not consider it.

Example Let's study the following code:

```
procedure max(a : array of n integers)
    max := a[1]
    for i := 2 to n
        if max < a[i] then max := a[i]
    return max
```

The first observation we can make is that the loop is executed $c(n) = n - 1$ times (where c is the cost function). We are doing two comparisons at each iteration of the loop, one in the if, and one in the condition of the for loop. We are doing one last comparison at the end of the loop, to know we have to get out of it.
So, the total cost for input of size n is $2 \cdot c(n) + 1$. $c(n)$ is $\Theta(n)$, $2c(n)$ is $\Theta(n)$ and $2 \cdot c(n) + 1$ is $\Theta(n)$.
It is clear we have done quite too much work. In fact, we do not care about some extra operations "outside the loop", a constant number of operations "inside the loop" (constant when we modify n, not that differ at each iteration). In other words, we only had to count the loops.

9.3 Complexity of searching and sorting algorithms

Worst case complexity of linear search

As a reminder, here is the code for linear search:

```
procedure linearsearch(x: integer, a: array of n distinct integers)
    i := 1
    while(i <= n and x != a[i])
        i := i + 1
    if i <= n then location := i else location := 0
    return location
```

We have a loop which is executed at most $n+1$ times, so $c(n) = n$, which is $\Theta(n)$.

Worst case complexity of binary search

As a reminder, here is the code for binary search:

```
procedure binarysearch(x: integer, a: array of n increasing integers)
    i := 1
    j := n
    while i < j
        m := floor((i + j)/2)
        if x > a[m] then i := m + 1 else j := m
    if x = a[i] then location := i else location := 0
    return location
```

Let's assume that $n = 2^k$ for convenience (it eases our explanation, and does not change anything in fact). After the first iteration, our list is of length 2^{k-1}, then it becomes 2^{k-2}, and so on until it is of length 1. So, this loop will be run $k+1$ times. However, we know that:

$$n = 2^k \implies k = \log_2(n)$$

Therefore, this algorithm has a cost function that is $\Theta(\log n)$. This has a slower growth than linear complexity, so it is faster than linear search (but it needs the array to be sorted before; adding the two costs makes binary search become slower if we will search something only once in the list (but if we search multiple times, then it becomes worth)).

Worst case complexity of bubble sort

As a reminder, here is the code for bubble sort:

```
procedure bubblesort(a: array of n >= 2 number)
    for i := 1 to n - 1
        for j := 1 to n - i
            if a[j] > a[j+1] then interchange(a[j], a[j+1])
```

We notice that the outer loop is executed $n-1$ times. The inner loop is executed $n-1$ times when $i=1$, $n-2$ times when $i=2$, and so on until 1 time when $i = n-1$.
So, in total, the inner loop is executed

$$(n-1) + (n-2) + \ldots + 1 = \frac{n(n-1)}{2} \text{ times}$$

which is a polynomial of degree 2.
Therefore, we deduce that the time complexity of this algorithm is $\Theta\big(n^2\big)$.

Worst case complexity of insertion sort

As a reminder, here is the code for insertion sort:

```
procedure insertionsort(a: array of n >= 2 numbers)
    for j := 2 to n
        i := 1
        while a[j] > a[i] and i < j
            i := i + 1
        m := a[j]
        for k := 0 to j - i - 1
            a[j - k] := a[j - k - 1]
        a[i] := m
```

We notice that the outer loop is executed $n-1$ times, the first inner loop is executed $n-1+\ldots+1$ times, and the second inner loop is also executed $n-1+\ldots+1$ times.

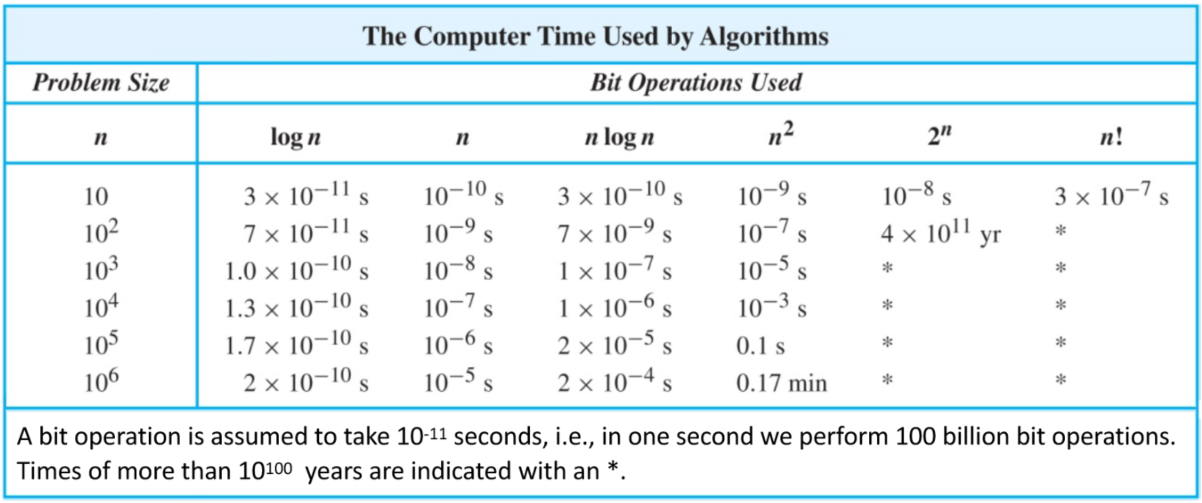
So, we deduce that the complexity of this algorithm is $\Theta\left(n^2\right)$, the same as bubble sort. We will see later that there exists other sorting algorithms which are $\Theta(n\log(n))$.

9.4 Understanding complexity

Note Here is the commonly used terminology for the complexity of algorithms:

Complexity	Terminology
$\Theta(1)$	Constant complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n\log n)$	Linearithmic complexity
$\Theta\left(n^b\right)$	Polynomial complexity
$\Theta(b^n)$, $b > 1$	Exponential complexity
$\Theta(n!)$	Factorial complexity

Tuesday 9th November 2021 — **Lecture 15 : P = NP and induction**



The Computer Time Used by Algorithms

Problem Size	*Bit Operations Used*					
n	$\log n$	n	$n\log n$	n^2	2^n	$n!$
10	3×10^{-11} s	10^{-10} s	3×10^{-10} s	10^{-9} s	10^{-8} s	3×10^{-7} s
10^2	7×10^{-11} s	10^{-9} s	7×10^{-9} s	10^{-7} s	4×10^{11} yr	*
10^3	1.0×10^{-10} s	10^{-8} s	1×10^{-7} s	10^{-5} s	*	*
10^4	1.3×10^{-10} s	10^{-7} s	1×10^{-6} s	10^{-3} s	*	*
10^5	1.7×10^{-10} s	10^{-6} s	2×10^{-5} s	0.1 s	*	*
10^6	2×10^{-10} s	10^{-5} s	2×10^{-4} s	0.17 min	*	*

A bit operation is assumed to take 10^{-11} seconds, i.e., in one second we perform 100 billion bit operations. Times of more than 10^{100} years are indicated with an *.

Effect of complexity We realise that the gap between polynomial complexity and exponential is really gargantuous.
Even $O\left(x^2\right)$ already grows fast, and it is not very efficient.

Tractable problems Commonly, a problem is considered **tractable**, if there exists an algorithm, which can solve the problem in polynomial time. In other words, there must exist an algorithm which can, for input of size n, produce the result with $O\left(n^d\right)$ operations. We call this class of algorithms P for polynomial complexity. For large d and large inputs, it is often not so clear that the problem is really tractable in a practical sense (example: multiplying two very large matrices).
If such an algorithm does not exist, the problem is considered **intractable**.

Class NP The class **NP** consists of problems for which the correctness of a proposed solution can be verified in polynomial time. It is for example much easier to verify the solution to an equation, than finding it.
NP stands for **non-deterministic polynomial time**. The non-deterministic is here because it means we can try random solutions and until we find the answer. Thus, NP can always be solved in exponential time (we only need a polynomial time to verify an answer, so we can try all the solutions).
Nowadays, there is still an open question on this subject: we wonder if P = NP. In other words, we wonder if it is possible to solve problems of the NP class using polynomial class.

Knapsack Decision problem

Given a set S of n items, each item with a value and a weight, find the subset of items that exceeds a minimal value while fitting a maximal weight.
This is a NP problem. No polynomial algorithm is known, whereas checking correctness of the solution is really easy.

NP-complete problems

NP-complete porblems are problems in NP, such that if a polynomial algorithm is found for this problem, then all other problems in NP can also be solved in polynomial time.
For example, the knapsack decision problems and the 3-SAT are NP-complete.

3-SAT

We want to know the satisfiability of propositional statements. As we saw, constructing a truth tables takes 2^n rows.
We will restrict a bit this problem to an easier one, but still NP-complete: 3-SAT.
We want to determine the satisfiability of formulas in conjunctive normal form, where each clause has at most 3 variables. For example, the following formula is included in the set of 3-SAT formulas:

$$(p \vee p \vee q) \wedge (\neg p \vee \neg q \vee \neg q) \wedge (\neg p \vee q \vee q)$$

Summary

We have seen the following types of algorithm:

- **Tractable problem:** There exists a polynomial time algorithm to solve this problem. These problems are said to belong to the **Class P**.
- **Class NP:** The solution can be checked in polynomial time, and any algorithm can be solved using exponential time. We wonder if it is possible to solve all NP problems using polynomial time.
- **NP Complete Class:** It is a subset of the NP class. If someone finds a polynomial-complexity algorithm for a member of the class, then it can be used to solve all the problems in the NP class.
- **Intractable Problems:** There does not exist a polynomial time algorithm to solve this problem.
- **Unsolvable Problem** There does not exist any algorithm to solve the problem (for example, the Halting Problem).

Chapter 10

Induction and recursion

10.1 Mathematical induction

Principle of mathematical induction

Mathematical induction is a tool that allows us to prove that a proposition is true for all integers.
To do that, we begin with a **basis step** by proving that $P(1)$ is true.
Then, we make an **inductive step**; we assume that $P(k)$ holds for an arbitrary integer k (**inductive hypothesis**), and we prove the proposition for $P(k+1)$.
After having done both those steps, we can conclude that the property is true for all n. What we have done is proving that all dominos fall, by proving that the first one fall, and that if one falls then the next one falls as well.

Example

Let's say we want to prove that $n < 2^n$ for all positive integers n.
Base step: We take $n = 1$, and we see that, indeed, $1 < 2$.
Inductive step: We assume that $1 \leq k < 2^k$. So, we have:

$$k + 1 \overset{\text{I.H.}}{<} 2^k + 1 \leq 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$$

Rule of inference

Mathematical induction can be expressed as the following rule of inference:

$$(P(1) \land \forall k(P(k) \to P(k+1))) \to \forall n P(n)$$

where the domain is the set of positive integers.

Important points

- In a proof by mathematical induction, we do not assume that $P(k)$ is true for all positive integers, only for some k. We then want to prove that $P(k+1)$ is true.
- We do not have to begin at 1. If we want to prove a property $\forall n \geq b$, where b is an integer, then we verify that $P(b)$ is true in our basis step, and not $P(1)$.

Validity

Mathematical induction is valid because of the well-ordering property (an axiom for the set of positive integers): every non-empty subset of the set of positive integers has a least element.
Mathematical induction is equivalent to the well ordering property.

Proof

The following proof is an extra, which we have not seen in class (so it will definitely not be in the exam).

Let P be a property for which $P(1)$ and $P(k) \to P(k+1)$ are true for all positive integers k. Let's suppose for contradiction that there exists a n such that $P(n)$ is false. Thus, the set S of positive integers for which $P(n)$ is false is non-empty. By the well-ordering property, we know that S has a least element; let's call that element m.

We know $m \neq 1$ since $P(1)$ is true by hypothesis. Thus, we can deduce that $m - 1$ is a positive integer ($m - 1 \geq 1$). Moreover, since $m-1<m$, and m was the least element of S, we know that $m-1\notin S$. In other words, we know that $P(m-1)$ is true. However, since $m-1$ is a positive integer and since we know that $P(k)\to P(k+1)$ is true for all positive integers k, then necessarily $P(m-1)\to P(m)$. Since $P(m-1)$ is true, then $P(m)$ must also be true, which contradicts our assumption that $P(m)$ is false.

We deduce that P is true for all $n\in\mathbb{N}$.

□

Example 1 We want to show that $2^n<n!$ for every integer $n\geq 4$.
Base step: We take $n=4$, and, indeed,

$$2^4=16<24=4!$$

Inductive step: We assume that $2^k<k!$ for some $k\geq 4$. We have:

$$2^{k+1}=2\cdot 2^k\overset{\text{IH}}{<}2k!<(k+1)k!=(k+1)!$$

Indeed, $k\geq 4\implies k+1>2$.

Example 2 We want to show that n^3-n is divisible by 3, for every positive integer n.
Base step: Taking $n=1$, we see that $1-1=0$ and, indeed, $3|0$.
Inductive step: We suppose that n^3-n is divisible by 3 (IH) for some n. We have:

$$(n+1)^3-(n+1)=n^3+3n^2+3n+1-n-1=\left(n^3-n\right)+3\left(n^2+n\right)$$

We know that $3|3\left(n^2+n\right)$ and, by our inductive hypothesis, $3|n^3-n$. So, 3 divides the sum.

Personal remark We can see that $n^3-n=n(n+1)(n-1)$. Thus, we could also have done a proof by cases:

$$n\equiv 0\mod 3\implies n(n+1)(n-1)\equiv 0(1)(-1)=0\mod 3$$

$$n\equiv 1\mod 3\implies n(n+1)(n-1)\equiv 1(2)(0)=0\mod 3$$

$$n\equiv 2\mod 3\implies n(n+1)(n-1)\equiv 2(0)(1)=0\mod 3$$

This gives more intuition on why this property is true (even though it is an example on how induction works, so intuition on the example may not be that useful).

Example 3 We want to show that, if S is a finite set with a non-negative number of elements, n, then S has 2^n subsets.
Base Step: We take $n=0$, so we have $S=\varnothing$. We see that:

$$P(\varnothing)=\{\varnothing\}\implies|P(\varnothing)|=1=2^0$$

as required.
Inductive step: We have $|S|=k+1$. Let's select an element $a\in S$ and let $T=S\setminus\{a\}$. We know that $|T|=k$ and thus, by our inductive hypothesis, $|P(T)|=2^k$.
We see that S contains two kinds of subsets: those that do not contain a (which is exactly T, so there are 2^k of those), and those that contain a (it's the exact same subsets, but with a added, so there are also 2^k of them (in the slides, there is a more detailed and formal proof on why this is true)). So we do have that:

$$2^k+2^k=2^{k+1}$$

Strong induction To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps. First, we show that $P(1)$ is true (nothing changed here). Then, we show that the following implication is true for all positive integers k:

$$P(1) \land P(2) \land \ldots \land P(k) \to P(k+1)$$

In other words, we prove that all dominos fall by proving that the first one falls, and that if all the ones behind a domino fall, then the latter falls as well.
Note that we can always use strong induction instead of mathematical induction, but there is no reason to use it if it is simpler to use mathematical induction. In fact, the principles of mathematical induction, strong induction, and the well-ordering property are equivalent.

Example 1 We want to show that every positive integer n can be written as a sum of distinct power of two, i.e. that there exists a set of integers $S = \{k_1, \ldots, k_m\}$ such that:

$$n = \sum_{j=1}^{m} 2^{k_j}$$

In other words, we want to show that every positive integer has a binary representation.
Base step: We take $n = 1$. We have:

$$\sum_{j=1}^{1} 2^0 = 1 \implies S = \{0\}$$

Inductive Step: We assume that $k = \sum_{j=1}^{m} 2^{k_j}$ for $\{k_1, \ldots, k_m\}$. If k is even, then we definitely know that 2^0 is not part of the sum, since it is the only possible odd term. In other words, we know that $0 \notin S$. So, by adding the element 0 to the set, taking $S' = S \cup \{0\}$:

$$\sum_{j=1}^{m} 2^{k_j} + 2^0 = k + 1$$

If k is odd, then $\frac{k+1}{2}$ is an integer. Therefore, we know that $\frac{k+1}{2}$ can be written as a sum of powers of 2 (we are using strong induction). In other words, there exists a set S such that:

$$\frac{k+1}{2} = \sum_{j=1}^{m} 2^{k_j} \implies k + 1 = \sum_{j=1}^{m} 2 \cdot 2^{k_j} = \sum_{j=1}^{m} 2^{k_j + 1}$$

We can thus construct $S' = \{k_1 + 1, \ldots, k_n + n\}$.

Example 2 We want to show that every set of lines in the plane, which do not have any parallel lines, need a common point.
Base step: We know that $P(2)$ is true. If two lines are not parallel, then they have a crossing point.
Inductive Step: Let's consider $k+1$ lines, $\ell_1, \ldots, \ell_{k+1}$. By our inductive hypothesis we know that $\ell_1, \ldots, \ell_k$ meet in a common point P_1, and $\ell_2, \ldots, \ell_{k+1}$ meet in a common point, P_2.
Let's assume for contradiction that $P_1 \neq P_2$. Then, lines $\ell_2, \ldots, \ell_k$ all go through P_1 and P_2, which is a contradiction since we know that they are not parallel. We thus deduce that our inductive step is true as well.
Mistake: Our proof by contradiction is wrong. If we only have three lines, we are saying that "all the lines between ℓ_2 and ℓ_2 go through P_1 and P_2 and since they are not parallel it is not possible", but, in this case, it makes no sens because we only have one line, ℓ_2.

Personal remark This false-proof is very similar to the one we saw in analysis, which proved that all the pencils are of the same colour, in any pencil case.

10.2 Recursively defined functions

Definition A **recursive** or **inductive** definition of a function f, which has a domain of non-negative integers, consists in two step. First, we define the **basis step**: we specify the value of the function at zero. Then, we define the **recursive step**: we give a rule for finding the function's value at an integer from its values at smaller integers. Note that a function $f(n)$ over natural numbers is a sequence $a_0, a_1, \ldots$ where $f(i) = a_i$.

Example 1 Let's say that f is defined by $f(0) = 3$, and:

$$f(n+1) = 2f(n) + 3$$

We can compute some values:

$$f(0) = 3, \quad f(1) = 2 \cdot 3 + 3 = 9, \quad f(2) = 2 \cdot 9 + 3 = 21, \quad \ldots$$

Example 2 The factorial function gets defined very well recursively:

$$f(n) = n! = \begin{cases} 1, & \text{if } n = 0 \\ f(n+1) = (n+1)f(n), & \text{else} \end{cases}$$

Fibonacci numbers The Fibonacci numbers are defined as follows:

$$f_0 = 0, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2}$$

We can compute some of the values:

$$f_0 = 0, \quad f_1 = 1, \quad f_2 = 1, \quad f_3 = 2, \quad f_4 = 3, \quad f_5 = 5$$

$$f_6 = 8, \quad f_7 = 13, \quad f_8 = 21, \quad f_9 = 34, \quad f_{10} = 56$$

We see that this function grows faster and faster. We'll see that it's, in fact, an exponential growth.

Wednesday 10th November 2021 — **Lecture 16 : Generalisation of induction and recursion**

Explanation of the Fibonacci sequence Leonardo Pisano Fibonacci considered the following problem: a young pair of rabbits (one of each gender) is placed on an island. A pair of rabbits does not breed until they are two months old. After a pair is 2 months old, it produces another pair each month.

The recursive relation that describes the population at month n is:

$$f_n = f_{n-1} + f_{n-2}$$

Indeed, the population at month n is the one of the month $n-1$ plus the pairs that are just born, which are equal to the number of reproductive pairs. The reproductive paire are those that are at least two months old and thus it is given by the number of rabbits at month $n-2$.

Property of the Fibonacci sequence Whenever $n \geq 3$, then:

$$f_n > \alpha^{n-2}, \quad \text{where } \alpha = \frac{1+\sqrt{5}}{2} = \varphi$$

We'll see later how to find this α without guessing.

Proof Let us use strong induction, but first, we can see that

$$\alpha^2 = \alpha + 1$$

Base step: For $n = 3$, we see that:

$$f_3 = 2 = \frac{1 + \sqrt{9}}{2} > \frac{1 + \sqrt{5}}{2} = \alpha = \alpha^{3-2}$$

Then, for $n = 4$:

$$f_4 = 3 = \frac{3 + \sqrt{9}}{2} > \frac{3 + \sqrt{5}}{2} = \alpha + 1 = \alpha^2 = \alpha^{4-2}$$

Inductive step: Since $\alpha^2 = \alpha + 1$:

$$a^{k-1} = \alpha^2 \alpha^{k-3} = (\alpha + 1)\alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$$

Thus:

$$f_{k+1} = f_k + f_{k-1} \overset{\text{IH}}{>} \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$$

□

10.3 Recursively defined sets and structure

Recursively defined sets and structure Recursion can be also used to define sets. **Recursive definitions of sets** have two parts. First, the basis step specifies an initial collection of elements. The recursive step gives the rules for forming new elements in the set from those already known to be in the set.
The **exclusion rule** states that only elements generated using the basis and recursive steps belong to this set.

Remark Previously, we either enumerated elements:

$$\{s_1, s_2, s_3\}$$

either used its properties:

$$\{x | P(x)\}$$

So, this gives us a third way.

Example 1 We can recursively define the set of natural numbers $\mathbb{N}$:

- Basis step: $0 \in \mathbb{N}$
- Recursive step: if n is in $\mathbb{N}$, then $n + 1$ is in $\mathbb{N}$.

Example 2 Let's recursively define a subset of integers $S3$:

- Basis step: $3 \in S3$
- Recursive step: if $x \in S3$ and $y \in S3$, then $x + y \in S3$.

Let's construct a sequence of "generations" for this set:

$$S3_0 = \{3\}, \quad S3_1 = \{3, 6\}, \quad S3_2 = \{3, 6, 9, 12\}$$

We understand that, in the end, we will get all the multiples of 3, even though this set is constructed in a weird way.

Strings Let's define the set Σ^* of strings over the alphabet Σ:

- Basis step: $\lambda \in \Sigma^*$ (λ is the empty string)

- Recursive step: if w is in Σ^* and x in Σ, then $wx \in \Sigma^*$ (where $w \cdot x = wx$ is the concatenation of the string w with the character x; it is the primary operation for strings).

Example 1 If $\Sigma = \{0, 1\}$, the strings in Σ^* are the set of all bit strings. We can construct the "generations" of this set:

$$\Sigma_0^* = \{\lambda\}, \quad \Sigma_1^* = \{\lambda, 0, 1\}, \quad \Sigma_2^* = \{\lambda, 0, 1, 00, 01, 10, 11\}, \quad \ldots$$

Example 2 Let $\Sigma = \{a, b\}$. We want to show that aab is in Σ^*.
Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, then $a \in \Sigma^*$.
Now, since $a \in \Sigma^*$ and $a \in \Sigma$, then $aa \in \Sigma^*$.
Finally, since $aa \in \Sigma^*$ and $b \in \Sigma$, then $aab \in \Sigma^*$.

Definition of well-formed formulae The set of **well-formed formulae** in propositional logic involving T, F, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ is recursively defined as:

- Basis step: T, F and s, where s is a propositional variable, are well-formed formulae.
- Recursive step: If E and F are well formed formule, then:

$$(\neg E), \quad (E \wedge F), \quad (E \vee F), \quad (E \rightarrow F), \quad (E \leftrightarrow F)$$

are well formed formulae. The parenthesis are important, so that there is no problem of precedence after multiple generations.

Example We can show using this definition that the following proposition is in the set of well-formed formulae:

$$((p \vee q) \rightarrow (q \wedge F))$$

However, we see that the following proposition is not well formed, since there are no parenthesis:

$$p \wedge q$$

10.4 Recursively defied functions on recursively defined sets

Definition We can define functions by recursion on recursively defined sets. We follow the definition of the set.
This is basically the same idea as recursive sequences, but instead we can have any basis step (which is a natural number for $\mathbb{N}$) and any recursive step (which is to explain how it goes from n to $n+1$ for $\mathbb{N}$).

Example 1 We want to find a recursive definition, $l(w)$, which gives the length of the string w.

- Base step: $l(\lambda) = 0$
- Recursive step: If $w \in \Sigma^*$ and $x \in \Sigma$, then :

$$l(wx) = l(w) + 1$$

Example 2 The **concatenation** of two strings w_1 and w_2, denoted by $w_1 \cdot w_2$, or $w_1 w_2$, is defined recursively as follows.
Let $w \in \Sigma^*$ be any string, and w_2 be the string we apply recursion on.

- Basis step: We take $w_2 = \lambda$: $w \cdot \lambda = w$.
- Recursive step: We take this definition as granted for some w_2, and we want to define it for $w_2 \cdot x$, where $x \in \Sigma$:

$$w \cdot (w_2 \cdot x) = (w \cdot w_2) \cdot x$$

10.5 Structural induction

Structural induction To prove a property of the elements of a recursively defined set, we use **structural induction**:

- **Basis step:** Show that the result holds for all elements specified in the basis step of the recursive definition.
- **Recursive step:** Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result hold for these new elements.

> *Remark* The validity of structural induction can be shown using the principle of mathematical induction.

> *Personal note* This is generalising the idea of induction over the natural numbers. Indeed, using the recursive definition of the natural numbers, we are using a natural number as the basis step, and we show that if the property is true for n, then it is also true for $n+1$.

Example 1 We want to prove that, if $x, y \in \Sigma^*$, then:

$$l(xy) = l(x) + l(y)$$

We want to show that $P(y) : l(xy) = l(x) + l(y)$ where x is fixed.

- **Base step:** We want to show $P(\lambda)$:

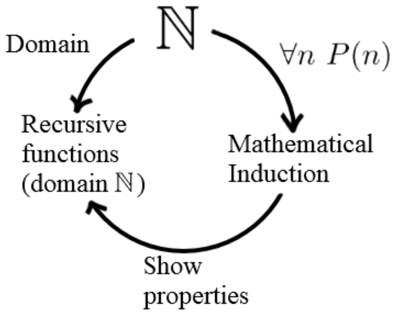
  $$l(x\lambda) = l(x) = l(x) + 0 = l(x) + l(\lambda)$$

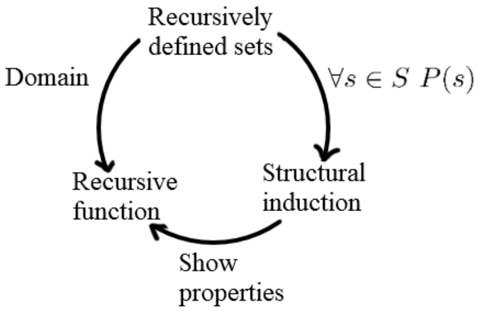
  by the recursive definition of concatenation and of $l(x)$.
- **Recursive step:** We assume that $P(y)$ is true, and we want to show that $P(ya)$ is true, where $a \in \Sigma$:

  $$l(x(ya)) \overset{\text{conc.}}{=} l((xy)a) \overset{\text{def } l}{=} l(xy) + 1 \overset{\text{IH}}{=} l(x) + l(y) + 1 \overset{\text{def } l}{=} l(x) + l(ya)$$

Example 2 We could show that every well-formed formula for compound propositions contains an equal number of left and right parenthesis.

Summary We generalised the idea of recursion and induction to any kind of recursively defined sets.





10.6 Recursive algorithms

Recursive algorithms definition

An algorithm is called **recursive** if it solves a problem by reducing it to an instance of the same problem with smaller input.
For the algorithm to terminate, the instance of the problem must eventually be reduced to some initial case for which the solution is known. (Don't forget the exit condition! ;))

WOW! HOW DID YOU GET LIKE THAT?
EVERY TIME I DO ONE PUSH-UP...
...I DO ONE PUSH-UP.
JESUS CHRIST.
SHEN COMIX

Recursive factorial algorithm

Here is an example of a recursive algorithm for computing $n!$ where n is a nonnegative integer:

```
procedure factorial(n):
    if n = 0 then return 1
    else return n*factorial(n-1)
```

Let's say we call `factorial(4)`, then it would call `4*factorial(3)`, which would call `4*(3*factorial(2))`, and so on:

```
factorial(4) =
4*factorial(3) =
4*(3*factorial(2)) =
4*(3*(2*factorial(1))) =
4*(3*(2*1)) =
4*(3*2) =
4*6 =
24
```

Many recursive calls get quickly inefficient in memory.

Recursive exponential algorithm

The following algorithm is an example of a recursive algorithm for computing a^n, where a is a nonzero real number and n is a nonegative integer.

```
procedure power(a, n):
    if n <= 0 then return 1
    else return a*power(a, n-1)
```

It is a really bad algorithm since $\Theta(n)$. The algorithm below is much better since it is $\Theta(\log(n))$.

```
procedure fast_power(a, n):
    if n <= 0 then return 1
    else a^(n & 1) * (fast_power(a, floor(n/2)))^2
```

We use `n & 1` to make a bitwise and between `n` and `1`, which is equivalent to take a modulo 2.

Recursive linear search

We can use the following recursive algorithm to do some linear search:

```
procedure recursive_linear_search(a: array of n real numbers, x: real
    number, i <= n: integer):
    if a[i] = x then return i
    else if i = n then return -1
    else return recursive_linear_search(a, x, i+1)
```

In the end, it does the same thing as linear search, but while executing it, we keep track of everything having happened before, so it is not very efficient.

Recursive binary search

Here is the recursive version of binary search:

```
procedure recursive_binary_search(a: array of n integers, x, i, j):
    m := floor((i + j)/2)
    if x = a[m] then return m
    else if(x < a[m] and i < m) then return recursive_binary_search(a,
         x, i, m-1)
    else if (x > a[m] and j > m) then return recursive_Binary_sarch(a,
         x, m+1, j)
    else return -1
```

The recursive version of this algorithm is not more efficient, however it is easier to ready.

Link between recursion and induction

Induction and recursion are different approaches to proving result and solving problems. They both rely on the ability of achieving the desired result for the smallest possible version of the problem. However, induction extends this ability to problems of any size, whereas recursion reduces a problem of any size to the smallest possible ones.

Proving correctness of recursive algorithms

We want to prove that the `power(a, n)` defined above is correct.

- Base step: `power(a, 0) = 1` which is good
- Inductive step: We assume that `power(a, k) =` a^k (IH):
  `power(a, k+1) =` $a \cdot$`power(a,k)` $= a \cdot a^k = a^{k+1}$

Recursion and iteration

A recursively defined function can be evaluated in two different ways:

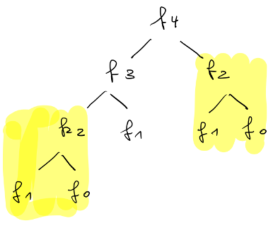
- **Recursively:** For a value, apply directly the recursive definition, until a base case is reached.
- **Iterative:** Start with a base case, and apply the recursive definition to compute the function for larger values.

Example

Let's say we want to compute the Fibonacci sequence using this algorithm:

```
procedure recursive_fibonacci(n):
    if n <= 1 then return n
    else return recursive_fibonacci(n-1) + recursive_fibonacci(n-2)
```

The problem with this algorithm is that we are computing multiple times some part, there are redundant computations (see picture below). The function calls twice itself each time, so the execution takes an exponential quantity of time.



The iterative version is much more efficient, it has a complexity of $\Theta(n)$.

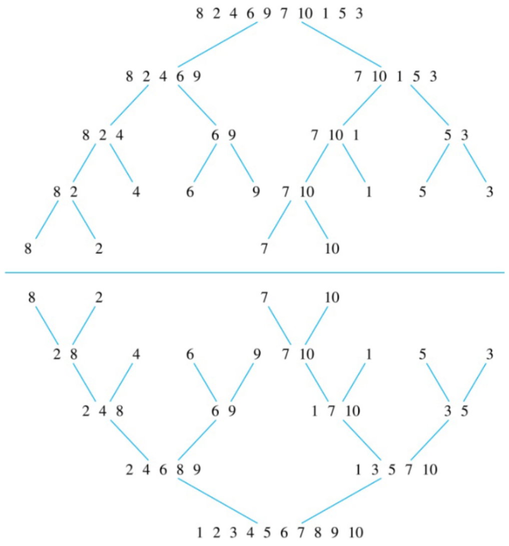
```
procedure iterative_fibonacci(n):
    if n = 0 then return 0
    else:
        previous := 0  // fibonacci(n-2)
        current := 1  // fibonacci(n-1)
        for i := 1 to n-1:
            next := previous + current
            previous := current
            current := next
        return current
```

Tuesday 16th November 2021 — **Lecture 17 : I have a proof too large to fit in the margin**

10.7 Merge sort

Recursive Sorting

We saw sorting algorithms that had $\Theta(n^2)$ complexity.
Merge sort is a sorting algorithm that performs significantly better. It works by iteratively splitting a list into two sublists of equal length, until each sublist has one element. At each step, a pair of sublists is successively merged into a list with the elements in increasing order. The process ends when all sublists have been merged.



Merge function

We use the following algorithm to merge two sorted lists into a sorted list:

```
procedure merge(a1, a2: sorted array)
    a := empty array
    while a1 and a2 are both nonempty
        remove the smaller amongst the first element of a1 and the one
             of a2 from its list
        put it at the end of a
    if one list is empty then remove all elements from the other list
        and append them to L.
    return a
```

The idea behind this function is that we can always consider the first element of both list, and pick the one that is the smallest, since they are both already sorted. The cost of merging two lists is at most `len(a1) + len(a2) - 1`. Let's say we have $1, 3, 5$ and $2, 4, 6$. We will pick the first element of the first list, then from the second list, and so on until there are 5 and 6 left. We need to make in total a comparison for each item added to the new list, except for the last one.
The best scenario is if we have something like 1 and $2, 3, 4, 5$. If all the elements of a list are smaller than the second one, then it will make the same number of comparisons as there are elements in this list, meaning `len(a1)`.

Pseudo code

Here is an example on how to write merge sort:

```
procedure mergesort(a: array of n elements)
    if n > 1 then:
        m := floor(n/2)
        a1 := a[1], a[2], ..., a[m]
        a2 := a[m+1], a[m+2], ..., a[n]
        return merge(mergesort(a1), mergesort(a2))
    else:
        return a
```

Complexity For simplicity, let's assume that n is a power of 2, say 2^m.

At each invocation of merge sort, the number of lists doubles and the length of the lists half. After m invocations there are $n = 2^m$ lists of length 1.

The merge procedure is now executed m times. At each execution, the length of lists doubles and their number halves. The cost of the merge procedure is the sum of the length of the lists minus 1.

Thus, the total cost is at most the following number of comparisons:

$$2^0\left(2^m-1\right)+2^1\left(2^{m-1}-1\right)+\ldots 2^{m-1}\left(2^1-1\right)=\sum_{k=1}^{m}\underbrace{2^{k-1}}_{\#\text{ merges}}\underbrace{\left(2^{m-k+1}-1\right)}_{\text{cost of merge}}$$

We can simplify our sum:

$$\sum_{k=1}^{m}2^{k-1}\left(2^{m-k+1}-1\right)=\sum_{k=1}^{m}2^m-\sum_{k=1}^{m}2^{k-1}=m2^m-2^m+1$$

We know that $n = 2^m \iff m = \log_2(n)$:

$$m2^m-2^m+1=\log_2(n)n-n-1$$

So, this algorithm has complexity $\Theta(n\log(n))$, i.e. linearithmic complexity. This is the best complexity that can yet be obtained for sorting, it is much better than $\Theta\left(n^2\right)$.

Chapter 11

Number theory

11.1 Division

History

Euclid wrote the most successful mathematics book ever "the elements" (it has over 1000 editions); Gauss said "mathematics is the queen of sciences, number theory is the queen of mathematics"; Wiles proved Fermat's last theorem 358 years after it was conjectures (someone please give bigger margins to Fermat) in 1958; Tao proved a famous theorem on arbitrarily long arithmetic processions of primes in 1975.
This is a very old and important subject.

Number theory and computer science

We use number theory in the representation and computation of integers, cryptography, coding, pseudo-random number generation, and so on.
Computing is also used to explore open problems in number theory, such as in the Collatz conjecture.

Definition of division

If a and b are integers, with $a \neq 0$, then ***a* divides *b*** if there exists an integer c such that $b = ac$. When a divides b, we say that a is a **factor** or **divisor** of b, and that b is a **multiple** of a.

> *Notation* The notation $a \mid b$ denotes a divides b. If a does not divide b, then we write $a \nmid b$.

> *Observation* We see that, if $a \mid b$, then:
>
> $$a \mid b \iff b = ak \iff \frac{b}{a} = k \in \mathbb{Z}$$
>
> So, $\frac{b}{a}$ is an integer if and only if $a \mid b$.

Properties of divisibility

1. If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$
2. If $a \mid b$, then $a \mid bc$ for all integers c
3. If $a \mid b$ and $b \mid c$, then $a \mid c$

> *Proof of property (1)* We can write $b = ak_1$ and $c = ak_2$, where k_1 and k_2 are integers. Therefore:
>
> $$b + c = ak_1 + bk_2 = a(k_1 + k_2)$$
>
> Since $k_1 + k_2$ is an integer, then $a \mid b + c$.

Corollary

If a, b, c are integers, where $a \neq 0$ such that $a \mid b$ and $a \mid c$, then:

$$a|mb + nc, \quad m, n \in \mathbb{Z}$$

Division algorithm

When an integer is divided by a positive integer, there is a quotient and a remainder. This is traditionally called the “division algorithm”, but it is in fact a theorem.

Theorem

If a is an integer and d is a positive integer, then there are unique integers q and r, with $0 \leq r < d$, such that:

$$a = dq + r$$

We can use the well-ordering of the natural numbers (and thus induction) to prove this theorem.

Terminology

If we have $a = dq + r$, then:

- d is called the **divisor**.
- a is called the **dividend**.
- q is called the **quotient**.
- r is called the **remainder**.

We write:

$$q = a \operatorname{div} d, \quad r = a \bmod d$$

div and mod are functions:

$$\operatorname{div} : \mathbb{Z} \times \mathbb{Z}_+ \mapsto \mathbb{Z} \quad \text{and} \quad \operatorname{mod} : \mathbb{Z} \times \mathbb{Z}_+ \mapsto \mathbb{N}$$

Example 1

We want to find the quotient and remainder when 101 is divided by 11. We see that 101 can be written as:

$$101 = 9 \cdot 11 + 2$$

Therefore, $q = 101 \operatorname{div} 11 = 9$ and $r = 101 \bmod 11 = 2$.

Example 2

Now, we want to divide -11 by 3. We see that:

$$-11 = -4 \cdot 3 + 1 \implies q = -11 \operatorname{div} 3 = -4 \text{ and } r = -11 \bmod 3 = 1$$

We have to be careful since dividing (positive) 11 by 3 is completely different:

$$11 \operatorname{div} 3 = 3, \quad 11 \bmod 3 = 2$$

11.2 Congruence

Definition of congruence

Let a and b are integers and m be a positive integer. We say that a is **congruent** to b modulo m if:

$$m \mid a - b$$

We note $a \equiv b \pmod{m}$ to say that a is congruent to b modulo m. We say that $a \equiv b \pmod{m}$ is a congruence, and that m is its modulus. If a is not congruent to b modulo m, we write $a \not\equiv b \pmod{m}$.
We say that $a \equiv b \pmod{m}$

Example

We wonder if 17 is congruent to 5 modulo 6:

$$17 - 5 = 12, \quad 6 \mid 12 \implies 17 \equiv 5 \pmod{6}$$

We can now see if 24 and 14 are congruent modulo 6:

$$24 - 14 = 10, \quad 6 \not\mid 10 \implies 24 \not\equiv 14 \pmod{6}$$

Theorem

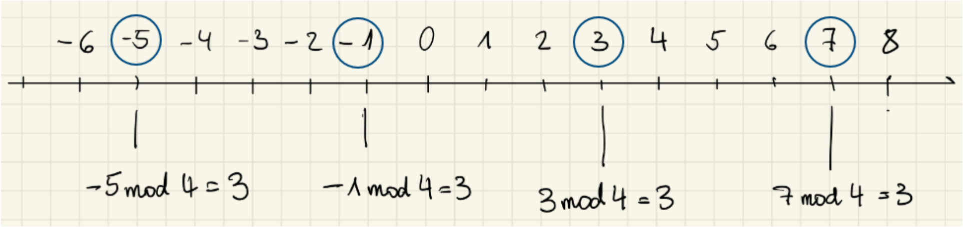
$a \equiv b \pmod{m}$ is an equivalence relation.

Notations We saw that the notations $a \equiv b \pmod{m}$ and $a \bmod m = b$ are different. The first one is a relation, the second one is a function.

Theorem 1 Let a and b be integers, and let m be a positive integer. We have:

$$a \equiv b \pmod{m} \iff a \bmod m = b \bmod m$$

Corollary Two integers are congruent mod m if and only if they have the same remainder when divided by m.



Theorem 2 Let m be a positive integer, and a, b be integers. We have:

$$a \equiv b \pmod{m} \iff \exists k \in \mathbb{Z} \text{ such that } a = b + km$$

Theorem 3 Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then:

$$a + c \equiv b + d \pmod{m} \quad \text{and} \quad ac \equiv bd \pmod{m}$$

Warning We have to beware of the fact that it does not work for powers. Indeed, we can have the following counter-example:

$$2 \equiv -1 \pmod{3} \text{ and } 3 \equiv 0 \pmod{3}$$

However:

$$3 \nmid 7 \implies 3 \nmid 2^3 - (-1)^0 \implies 2^3 \not\equiv (-1)^0 \pmod{3}$$

Proof of multiplicativity We have

$$a = b + k_1 m \quad \text{and} \quad c = d + k_2 m$$

So:

$$ac = (b + k_1 m)(d + k_2 m) = bd + k_1 md + k_2 mb + k_1 k_2 m = bd + \overline{k} m$$

Which allows us to conclude:

$$ac \equiv bd \pmod{m}$$

So, this operation is very robust under elementary operations.

Example If we know that $7 \equiv 2 \pmod{5}$ and $11 \equiv 1 \pmod{5}$, then we know that:

$$77 \equiv 2 \pmod{5}$$

Corollary Adding an integer to both sides of a valid congruence preserves validity:

$$a \equiv b \pmod{m} \implies a + c \equiv b + c \pmod{m}, \quad \forall c \in \mathbb{Z}$$

Multiplying both sides of a valid congruence by an integer preserves validity:

$$a \equiv b \pmod{m} \implies c \cdot a \equiv c \cdot b \pmod{m}, \quad \forall c \in \mathbb{Z}$$

Putting both sides of a valid congruence to an integer power preserves validity:

$$a \equiv b \pmod{m} \implies a^c \equiv b^c \pmod{m}, \quad \forall c \in \mathbb{Z}$$

Proof We can use the last theorem for the two first points. Then, for the third one, we can see that:

$$a \equiv b \pmod{m} \implies aa \equiv bb \pmod{m} \implies aaa \equiv bbb \pmod{m}$$

Repeating this idea n times:

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

Warning We have to beware of the fact that we can neither divide by an integer, nor take a nth-root:

$$14 \equiv 8 \pmod{6} \quad \text{but} \quad 7 \not\equiv 4 \pmod{6}$$

$$16 \equiv 9 \pmod{7} \quad \text{but} \quad 4 \not\equiv 3 \pmod{7}$$

Dividing by an integer is a bit more special, because in some specific cases it will work ($ac \equiv bc \pmod{m} \implies a \equiv b \pmod{m}$ if c and m are coprimes).

Property of congruences We wonder if:

$$a \equiv b \pmod{pq} \implies a \equiv b \pmod{p}$$

We have:

$$a \equiv b \pmod{pq} \implies a = b + kpq \implies a = b + (kq)p \implies a \equiv b \pmod{p}$$

So yes. Now we may wonder if the reciprocal holds, but we can find a counterexample:

$$1 \equiv 5 \pmod{2}, \quad 1 \equiv 5 \pmod{4} \quad \text{but} \quad 1 \not\equiv 5 \pmod{8}$$

11.3 Modular arithmetic

Corollary Let m be a positive integer, and let $aandb$ be integers. Then:

$$(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$

$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$

Proof We are using the fact that:

$$a \equiv a \bmod m \pmod{m}$$

Example This corollary allows us to simplify some computations:

$$280 \bmod 6 = (28 \bmod 6 \cdot 10 \bmod 6) \bmod 6 = (4 \cdot 4) \bmod 6 = 16 \bmod 6 = 4$$

Definitions Let $\mathbb{Z}_m$ be the set of nonnegative integers less than m:

$$\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$$

The **addition modulo m** $+_m$ is defined as:

$$a +_m b = (a+b) \bmod m$$

The **multiplication modulo *m*** $\cdot_m$ is defined as:

$$a \cdot_m b = (a \cdot b) \bmod m$$

Example For example:

$$7 +_{11} 9 = 5, \quad 7 \cdot_{11} 9 = 8$$

Tables We can draw tables for our operations. For example, in $\mathbb{Z}_3$:

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\cdot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

We notice that we get $1 \cdot_3 1 = 1$ and $2 \cdot_3 2 = 1$. So, for $n \neq 0$, numbers have a multiplicative inverse on this set.

Wednesday 17th November 2021 — **Lecture 18 : Basis, arithmetic, and primes**

More tables Let's look at $\mathbb{Z}_4$:

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

We see that we do not have a multiplicative inverse for every non-zero number: there is no x such that $2 \cdot x = 1$.
In fact, $\mathbb{Z}_m$ has a multiplicative inverse if and only if m is prime.

Arithmetic modulo m The operations $+_m$ and $\cdot_m$ satisfy many of the usual properties, the same way as ordinary addition and multiplication,:

- Closure: $a + b \in \mathbb{Z}_m$, $a \cdot b \in \mathbb{Z}_m$
- Commutativity: $a + b = b + a$, $a \cdot b = b \cdot a$
- Associativity: $a + (b + c) = (a + b) + c$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- Distributivity: $a \cdot (b + c) = a \cdot b + a \cdot c$
- Existence of the additive identity: $x + 0 = x$
- Existence of the multiplicative identity, $x \cdot 1 = x$
- Existence of the additive inverse: $x + (-x) = 0$

The only problem is the existence of multiplicative inverse.
In the terminology of abstract algebra, $\mathbb{Z}_m$ with $+_m$ is **commutative group**, and $\mathbb{Z}_m$ with $+_m$ and $\cdot_m$ is a commutative ring.

11.4 Integer representation

Introduction In general, we use **decimal** — or **base 10** — notation to represent integer.
For example, when we write 965, we mean:

$$9 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0$$

In fact, we could use any base b, where b is a positive integer greater than 1. The most important bases for computing are base 2 (binary), base 8 (octal), and base 16 (hexadecimal).
In the ancient times, Mayans used base 20 (number of fingers and toes), and Babylonians used base 60 (it is why there are 60 seconds in one minute, or 360° is a full circle).

Base b representation Let b be a positive integer greater than 1, n be a positive integers. Then, there exists $k \in \mathbb{N}_0$ and $a_0, \ldots, a_k \in \mathbb{N}_0$, and $a_k \neq 0$, such that:

$$n = a_k b^k + \ldots + a_1 b + a_0$$

The coefficients $a_0, \ldots, a_k$ are called the base-b digits of the representation, and it is denoted

$$(a_k a_{k-1} \ldots a_1 a_0)_b$$

We usually omit the subscript 10 for base 10 expansions.

Proof We can use induction to prove this theorem (we did a similar proof to prove that the binary expansion is unique earlier).

Binary expansion

Most computers represent numbers and do arithmetic with binary (since, in their circuits, electricity either flows or not). In these expansions, the only digits used are 0 and 1.
For example:

$$(101011111)_2 = 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 351$$

Octal expansion

The octal expansion uses the digits $\{0, 1, 2, 3, 4, 5, 6, 7\}$.
For example:

$$(7016)_8 = 7 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 6 \cdot 8^0 = 3598$$

Hexadecimal expansion

For this expansion, we need 16 digits, so we use letters to represent numbers 10 through 15. Thus, we use the digits $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.
For example:

$$(2AE0B)_{16} = 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16^1 + 11 \cdot 16^0 = 175627$$

Obtaining a base *b* expansion

Let's say we want to get the base 2 expansion of $(11)_{10}$. We have:

$$11 = 8 + 2 + 1 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

Obtaining this last equality may be very hard through trial and error, here it is just for more illustration (if we have this result, we are done, we would not need the following algorithm). We see that:

$$11 \bmod 2 = 1, \quad 11 \operatorname{div} 2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$$

So, we can do it again:

$$5 \bmod 2 = 1, \quad 5 \operatorname{div} 2 = 1 \cdot 2^1 + 0 \cdot 2^0 = 2$$

$$2 \bmod 2 = 0, \quad 2 \operatorname{div} 2 = 1 \cdot 2^0 = 1$$

$$1 \bmod 2 = 1, \quad 1 \operatorname{div} 2 = 0$$

And that's where the process ends. We can now read the numbers from bottom to top:

$$(11)_{10} = (1011)_2$$

It may be easier to write our numbers in a table:

$x_n = x_{n-1} \operatorname{div} 2$	11	5	2	1	0
$x_n \bmod 2$	1	1	0	1	

We can then read the last line from right to left.

Pseudocode

Here is the pseudocode for this algorithm:

```
procedure base_b_expansion(n, b > 1)
    q := n
    k := 0
    while(q != 0)
        a[k] := q mod b
        q := q div b
        k := k + 1
    return {a[k-1], ..., a[1], a[0]}  // base b expansion of n
```

Example

We want to find the octal expansion of $(12345)_{10}$. Successively dividing by 8 gives:

$$12345 = 8 \cdot 1543 + 1$$

$$\begin{aligned} 1543 &= 8 \cdot 192 + 7 \\ 192 &= 8 \cdot 24 + 0 \\ 24 &= 8 \cdot 3 + 0 \\ 3 &= 8 \cdot 0 + 3 \end{aligned}$$

So, $(12345)_{10} = (30071)_8$.

Comparison of hexadecimal, octal, and binary representations

We can draw the two following tables:

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Each octal digit corresponds to a block of 3 binary digits, and each hexadecimal digit corresponds to a block of 4 binary digits. Thus, to do conversion between binary and hexadecimal, and binary and octal (and thus octal and hexadecimal), we don't need to do all the computations, only to look up in the table. For example, converting a number in base 2, we only need to group numbers of 3:

$$(011\ 111\ 010\ 111\ 100)_2 = (37274)_8$$

Similarly, for hexadecimal:

$$(0011\ 1110\ 1011\ 1100)_2 = (3EBC)_{16}$$

Hexadecimal is very useful to shorten binary representation, to make them more readable. This is why private keys are written in hexadecimal, for example: they represent binary numbers, but they are made smaller. This is also why the lowest level programming is hexadecimal and not binary.

11.5 Arithmetic with base b expansions

Addition Let's recall how we add numbers. Let's say we want to add $a = (1110)_2$ and $b = (1011)_2$. We can do column addition:

$$\begin{array}{ccccc}
{\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\
 & 1 & 1 & 1 & 0 \\
+ & 1 & 0 & 1 & 1 \\
\hline
1 & 1 & 0 & 0 & 1
\end{array}$$

Let's try to extract the algorithm behind column addition. We see that, first, we begin by the right most elements. Moreover, we may have a carry. We have:

$$\begin{array}{lll}
a_0 + b_0 = 1 & c_1 = 0 & s_0 = 1 \\
a_1 + b_1 + c_0 = 1 + 1 + 0 = 10 & c_2 = 1 & s_1 = 0 \\
a_2 + b_2 + c_1 = 1 + 0 + 1 = 10 & c_3 = 1 & s_2 = 0 \\
a_3 + b_3 + c_2 = 1 + 1 + 1 = 11 & c_4 = 1 & s_3 = 1 \\
 & & s_4 = c_4 = 1
\end{array}$$

So, completely generally, we can deduce that, if we have $a_i + b_i + c_{i-1}$ then:

$$c_i = \left\lfloor \frac{a_i + b_i + c_{i-1}}{2} \right\rfloor \quad \text{and} \quad s_i = a_i + b_i + c_{i-1} - 2c_i$$

Binary addition of integers Here is pseudocode for adding two binary numbers:

```
procedure add(a, b: array of n elements defining binary expansions
    ):
c := 0
for j := 0 to n - 1:
    d := floor(a[j] + b[j] + c)/2
    s[j] := a[j] + b[j] + c - 2d
    c := d
s[n] := c
return {s[0], s[1], ..., s[n]}  // We could return s, but I think
    this is clearer
```

The number of additions of bits used by the algorithm to add two d-bit integers is $O(d)$ (note that d is not the number, but its number of digits).

Utility It might seem ridiculous to define addition using addition, multiplication and division. However, using this algorithm, we only need to define addition, division and multiplication for 1-bit numbers, and then we can use it for all possible numbers. Those definitions are hard-wired in the CPU.

Multiplication Let's say we want to multiply $a = (110)_2$ and $b = (101)_2$. Again, let's look at a column product:

$$\begin{array}{cccccc}
 & & & 1 & 1 & 0 \\
\cdot & & & 1 & 0 & 1 \\
\hline
 & & & 1 & 1 & 0 \\
 & & 0 & 0 & 0 & \bullet \\
+ & 1 & 1 & 0 & \bullet & \bullet \\
\hline
 & 1 & 1 & 1 & 1 & 0
\end{array}$$

We are using two observations to do that. We see that:

$$a \cdot b = a \cdot \left(b_k 2^k + \ldots b_0\right) = a b_k 2^k + a b_{k-1} 2^{k-1} + \ldots + a b_0 2^0$$

First, if $a = 2^j$, then we are adding j zeros at the end (those are the $\bullet$ in the column product). Second, we see that we are summing "partial product" (corresponding to the second step in the column product). For example, with our previous values:

$$a \cdot b_0 2^0 = (110)_2 \cdot 1 \cdot 2^0 = (110)_2$$

$$a \cdot b_1 2^1 = (110)_2 \cdot 0 \cdot 2^1 = (0000)_2$$
$$a \cdot b_2 2^2 = (110)_2 \cdot 1 \cdot 2^2 = (11000)_2$$

And then:

$$(110)_2 + (0000)_2 + (11000)_2 = (11110)_2$$

Algorithm Here is the pseudocode for multiplying two positive integers:

```
procedure multiply(a, b: arrays of n elements representing a positive
    integer)
    c := empty array of n elements  // partial products
    for j := 0 to n-1
        if b == 1 then c[j] := a with j zeros appended
        else c[j] := 0

    p := 0
    for j := 0 to n-1
        p := p + c[j]
    return p
```

This algorithm uses $O\left(d^2\right)$ additions, where d is the number of bits of the factors.

> *Utility* Again, as for addition, we only need to know 1-bit multiplication by 0 and by 1 to write this algorithm. We can then call the sum algorithm.

Binary modular exponentiation Notably in cryptography, it is sometimes important to be able to find $b^n \bmod m$ efficiently, where b, n and m are large integers.
Let's use the binary expansion of n to compute b^n:

$$b^n = b^{a_{k-1}2^{k-1} + \ldots + a_0} = \left(b^{2^{k-1}}\right)^{a_{k-1}} \cdot \ldots \cdot (b)^{a_0}$$

We thus see that, in order to compute b^n, we only need to be able to compute the values of:

$$b, \quad b^2, \quad b^4 = \left(b^2\right)^2, \quad b^8 = \left(b^4\right)^2, \quad \ldots, \quad b^{2^{k-1}}$$

Example Let's say we want to compute 3^{11}. First, let's not that $(11)_{10} = (1011)_2$. So:

$$3^{11} = 3^{1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0} = 3^{1\cdot 2^3} \cdot 3^{0\cdot 2^2} \cdot 3^{1\cdot 2^1} \cdot 3^{1\cdot 2^0} = 3^{2^3} \cdot 3^2 \cdot 3$$

And, we see that the computations begin to be very hard. Computing 3^1 to 3^4 is fine, but $3^8 = 81^2$ begins to be hard to be done by hand.

Applying the modulus Let's compute 3^{11} mod 5. Similarly:

$$3^{11} = 3^8 \cdot 3^2 \cdot 3$$

Now:

- $3^1 \bmod 5 = 3$
- $3^2 \bmod 5 = 4$
- $3^4 \bmod 5 = \left(3^2\right)^2 \bmod 5 = \left(3^2 \bmod 5\right)^2 \bmod 5 = 4^2 \bmod 5 = 1$
- $3^8 \bmod 5 = \left(\left(\left(3^2\right)^2\right)^2\right) \bmod 5 = 1^2 \bmod 5 = 1$

So:

$$3^{11} \bmod 5 = 3^8 3^2 3 \bmod 5 = \left(3^8 \bmod 5 \cdot 3^2 \bmod 5 \cdot 3^1 \bmod 5\right) \bmod 5 = 1{\cdot}4{\cdot}3 \bmod 5 = 2$$

So, even though 3^{11} is very large, modular arithmetic allows us to get those computations much more easily.

11.6 Prime numbers

Definition

A positive integer p greater than 1 is called **prime** if the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called **composite**.

Examples

For example, 7 is prime but 6 is composite.

The Fundamental Theorem of Arithmetic

If n is an integer greater than 1, then n can be written uniquely as a product of primes.

Examples

$999 = 3^3 \cdot 37$, $1024 = 2^{10}$, $641 = 641$

Proof of existence

Let's first prove by strong induction that n can be written as a product of primes (which may or may not be unique).

Basis step: $P(2)$ is true since 2 is a prime.

Inductive step: The inductive hypothesis is that $P(j)$ is true for all integers j with $2 \leq j \leq k$. To show $P(k+1)$, we consider two cases. If $k+1$ is prime, then $P(k+1)$ is true. Else, $k+1$ is composite and can thus be written as a product of two positive integers a, b where $2 \leq a \leq b < k+1$. By the inductive hypothesis, a and b can be written as a product of primes and therefore $k+1$ can also be written as the product of those primes.

Proof of unicity

Let's suppose for contradiction that there exists a $n \geq 2$ for which that there exists two distinct arrays of primes such that:

$$n = p_1 p_2 \ldots p_i = q_1 q_2 \ldots q_j$$

Since those two products are equal, we can simplify the primes numbers that are on both sides of the equation. Now, it means that there are no primes that are both in the left hand side and in the right hand side. Moreover, both sides still have primes (since $n \neq 1$, and the arrays are distinct).

So, let's consider p_m, the first element on the left hand side that has no been simplified previously. Since we know that prime numbers can only be divided by one and by themselves, we know that none of the number on the right hand side is divisible by p_m.

However, we know that if $a \nmid b$ and $a \nmid c$, then $a \nmid bc$, so, the right hand side is not divisible by p_m. However, since the left hand side is divisible by p_m, this is a contradiction.

□

Theorem (Trial division)

If n is a composite integer, then n has a prime divisor less than or equal to $\sqrt{n}$.

Usefulness

If we want to know if a number is prime, we thus only need to check for primes less than or equal to $\sqrt{n}$, which is much smaller than n.

Proof

By hypothesis, n is composite. Thus, it can be written as $n = ab$, where

$$2 \leq a, b \leq n$$

Let's assume that both numbers are greater than $\sqrt{n}$. In other words, $a > \sqrt{n}$ and $b > \sqrt{n}$. Thus:

$$n = ab > \sqrt{n}\sqrt{n} = n$$

Which is a contradiction. We thus know that $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$. Without loss of generality (WLOG), let's pick $a \leq \sqrt{n}$. So, either a is a prime, or a has a prime factor $\leq \sqrt{n}$. In both cases, we know our theorem holds.

□

Sieve of Eratosthenes The sieve of Eratosthenes can be used to find all primes not exceeding a specified positive integer.

Here is its idea. Let's write numbers from 2 to 100. The first number is a prime (it's 2), so we remove all the integers (other than 2) that are divisible by 2. Then, the first element after 2 which has not been removed is not a prime, and we can repeat those steps until all numbers have either been removed or seen as prime.

There is a nice animation on Wikipedia that explains this algorithm:

`https://commons.wikimedia.org/wiki/File:Sieve_of_Eratosthenes_animation.svg`

Infinitude of primes (Euclid's theorem) There are infinitely many primes.

Proof This proof was originally done by Euclid.

Let's assume for contradiction that there are $n \in \mathbb{N}$ primes: $p_1, \ldots, p_n$ are all the existing prime. Let's take the following number:

$$q = p_1 \cdot \ldots \cdot p_n + 1$$

We notice that none of $p_1, \ldots, p_n$ can divide q. Indeed, else:

$$p_i \mid q \implies q = kp_i = p_1 \cdot \ldots \cdot p_n + 1 \implies kp_i + p_1 \cdot \ldots \cdot p_n = 1$$

p_i can divide the left hand side but not the right hand side. This is a contradiction.

We deduce that q cannot be prime (since it is greater than all the primes), and it cannot be composite (since it would have a prime factor that is not in $p_1, \ldots, p_n$). This is a contradiction.

□

Tuesday 23rd November 2021 — **Lecture 19 : Euclid is the best, and we learn to count**

11.7 GCD and LCM

Definition of GCD Let a and b be integers, not both zero. The largest integer d such that $d \mid a$ and $d \mid b$ is called the **greatest common divisor** (GCD) of a and b. It is denoted by $\gcd(a, b)$.

Example We have:

$$\gcd(24, 36) = 12, \quad \gcd(17, 22) = 1$$

Definition of relatively prime The integers a and b are **relatively prime** if $\gcd(a, b) = 1$.

The integers $a_1, \ldots, a_n$ are **pairwise relatively prime** if $\gcd(a_i, a_j) = 1$ for $1 \leq i < j \leq n$.

Example

We can verify that $10, 17$ and 21 are pairwise relatively prime:

$$\gcd(10,17) = 1, \quad \gcd(10,21) = 1, \quad \gcd(17,21) = 1$$

However, 10, 19 and 24 are not pairwise relatively prime since:

$$\gcd(10,24) = 2 \neq 1$$

Finding the GCD

Let's write a and b as their prime factorisation:

$$a = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_n^{a_n}, \quad b = p_1^{b_1} \cdot p_2^{b_2} \cdot \ldots \cdot p_n^{b_n}$$

where each exponent is a nonnegative integer. We can compute the GCD the following way:

$$\gcd(a,b) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdot \ldots \cdot p_n^{\min(a_n,b_n)}$$

However, finding the prime factorisation of a number is a very complicated problem. It is not polynomial (and this is the basis of many cryptography algorithm).

Definition of LCM

The **least common multiple** (LCM) of two positive integers a and b is the smallest positive integer that is divisible by both a and b. It is denoted by $\operatorname{lcm}(a,b)$.
We can also find it using prime factorisations:

$$\operatorname{lcm}(a,b) = p_1^{\max(a_1,b_1)} \cdot p_2^{\max(a_2,b_2)} \cdot \ldots \cdot p_n^{\max(a_n,b_n)}$$

Example

Let's find the prime factorisation of 120 and 500:

$$120 = 2^3 \cdot 3 \cdot 5 = 2^3 \cdot 3^1 \cdot 5^1, \quad 500 = 2^2 \cdot 5^3 = 2^2 \cdot 3^0 \cdot 5^3$$

Then:

$$\gcd(120,500) = 2^2 \cdot 3^0 \cdot 5^1 = 4 \cdot 5 = 20$$

$$\operatorname{lcm}(120,500) = 2^3 \cdot 3^1 \cdot 5^3 = 8 \cdot 3 \cdot 125 = 3000$$

Theorem

Let a and b be positive integers. Then:

$$a \cdot b = \gcd(a,b) \cdot \operatorname{lcm}(a,b)$$

Proof

This proof relies on the fact that:

$$a + b = \max(a,b) + \min(a,b)$$

Thus:

$$\begin{aligned}
&\gcd(a,b) \operatorname{lcm}(a,b) \\
&= \left(p_1^{\min(a_1,b_1)} \cdot \ldots \cdot p_n^{\min(a_n,b_n)}\right)\left(p_1^{\max(a_1,b_1)} \cdot \ldots \cdot p_n^{\max(a_n,b_n)}\right) \\
&= p_1^{\min(a_1,b_1)} p_1^{\max(a_1,b_1)} \cdot \ldots \cdot p_n^{\min(a_n,b_n)} p_n^{\max(a_n,b_n)} \\
&= p_1^{\min(a_1,b_1)+\max(a_1,b_1)} \cdot \ldots \cdot p_n^{\min(a_n,b_n)+\max(a_n,b_n)} \\
&= p_1^{a_1+b_1} \cdot \ldots p_n^{a_n+b_n} \\
&= p_1^{a_1} \cdot p_1^{b_1} \cdot \ldots \cdot p_n^{a_n} \cdot p_n^{a_n} \\
&= p_1^{a_1} \cdot \ldots \cdot p_n^{a_n} \cdot p_1^{b_1} \cdot \ldots \cdot p_n^{b_n} \\
&= a \cdot b
\end{aligned}$$

Observation

If we have $a > b$ and $r = a \bmod b$, then:

$$\gcd(a,b) = \gcd(r,b)$$

Indeed, we know that $r = a \bmod b$ is equivalent to $a = q \cdot b + r$. We want to show that:

$$d \mid a \text{ and } d \mid b \iff d \mid b \text{ and } d \mid r$$

First, let's suppose that $d \mid a$ and $d \mid b$. Divisibility preserves addition, so

$$d \mid a - qb = r \implies d \mid r$$

Now, let's suppose that if $d \mid r$ and $d \mid b$:

$$d \mid q \cdot b + r = a \implies d \mid a$$

So, we have indeed shown that:

$$d \mid a \text{ and } d \mid b \iff d \mid b \text{ and } d \mid r$$

Example

Let's say we want to compute $\gcd(287, 91)$:

$$\gcd(287, 91) = \gcd(14, 91) = \gcd(91, 14) = \gcd(7, 14) = \gcd(14, 7) = \gcd(0, 7) = 7$$

Indeed, using long division we can see that:

$$287 = 3 \cdot 91 + 14$$

$$91 = 6 \cdot 14 + 7$$

$$7 = 2 \cdot 7 + 0$$

Euclidean algorithm

This algorithm is called the Euclidean algorithm (it was invented by Euclid). Iteratively, it is written:

```
procedure gcd_iterative(a > b: positive integers)
    x := a
    y := b
    while y != 0
        r := x mod y
        x := y
        y := r
    return x
```

Recursively, it looks much simpler:

```
procedure gcd_recursive(a > b: positive integers)
    if b == 0 then return a
    else return gcd(b, a mod b)
```

Proof

Let's say we want to compute $\gcd(a, b)$. So, we can take $r_0 = a$ and $r_1 = b$.

Then, we perform a division and we get $r_2 = r_0 \bmod r_1$. Clearly, $r_1 > r_2 \geq 0$.

Similarly, taking $r_3 = r_1 \bmod r_2$, then $r_2 > r_3 \geq 0$.

Since $r_i > r_{i+1} \geq 0$, then r decreases at least by 1 at each step, so there must be a step at which it is equal to 0. Let's call this $r_{n+1} = r_{n-1} \bmod r_n$, where $r_{n+1} = 0$. Then:

$$\gcd(a, b) = \gcd(r_0, r_1) = \ldots = \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n$$

Lamé's theorem

Let a and b be positive integers with $a \geq b$.

The number of divisions used by the Eucliden algorithm to find $\gcd(a, b)$ is less than or equal to five times the number of decimal digits in b.

In other words, the Euclidean algorithm has a complexity of $O(\log b)$ (careful, it is a big-O, not a big-Θ).

11.8 More facts about number theory (not in the exam)

Distribution of primes

Let's define $\pi(x)$ be the number of primes not exceeding x.
The prime number theorem states that the following function approaches 1 as x tends toward infinity:

$$\frac{\pi(x)}{\frac{x}{\ln(x)}}.$$

This implies that the odds that a random positive integer less than n is prime are approximately of:

$$\frac{1}{\ln(x)}.$$

Primes and arithmetic progressions

We wonder if there are long arithmetic progressions made up entirely of primes. For example, the following one consists of five primes:

$$5, 11, 16, 23, 29$$

We can also find one of ten primes:

$$199, 409, 619, 820, 1039, 1249, 1459, 1669, 1879, 2089$$

In the 1930s, Paul Erdös conjectured that for every positive integer $n > 1$, there is an arithmetic progression of length n made up entirely of primes. This was proven 70 years later, in 2006, by Ben Green and Terrence Tau.

Mersene primes

Prime numbers of the forme $2^p - 1$, where p is prime, are called Mersene primes. For example, the following numbers are Mersene primes.

$$2^2 - 1 = 3, \quad 2^3 - 1 = 7, \quad 2^5 - 1 = 37, \quad 2^7 - 1 = 127$$

However, all numbers of this form are not primes, for example the following number isn't prime (so it isn't a Mersene prime):

$$2^{11} - 1 = 2047 = 23 \cdot 89$$

The advantage of such primes is that there exists a very efficient way to verify whether numbers of the form $2^p - 1$ are primes. Thus, the largest primes nowadays are Mersene primes. In mid 2018, the largest known prime — which has 25 million decimal digits — was:

$$2^{82\,589\,933} - 1$$

Goldbach's Conjecture

In 1742 Goldbach conjectured that every even integer $n > 2$ is the sum of two primes.
It has been verified by computer for all positive integers up to $1.6 \cdot 10^{18}$, but no one has found a proof yet.

The Twin Prime Conjecture

We define twin primes to be a pair of primes that differ by 2. For example:

$$3, 5 \quad 5, 7 \quad 11, 13$$

It is conjectured that there are infinitely twin primes.
In 2018, the largest twin primes we found — which have 388 342 decimal digits — are:

$$2\,996\,863\,034\,895 \cdot 2^{1\,290\,000} \pm 1$$

Chapter 12

Counting

12.1 Basic rules

Introduction

We are asking ourselves some of the following questions:

- How many different passwords exists?
- How many outcomes does an experiment have?
- How many steps an algorithm performs?
- How many moves are possible in a game?

Counting is the subject of the mathematical discipline of **combinatorics**.

Short history

Mahavira (850 AD) provided formulas for permutations and combinations. Pascal (17^{th} century) developed Pascal's triangle. Euler (18^{th} century) was the first to use generating functions.

Product rule

Let A and B be two tasks, with respectively n_1 and n_2 ways to do them.
There are $n_1 \cdot n_2$ ways to do both tasks.

Example

When rolling two d-6 (dices with 6 faces), there is a total of $6 \cdot 6 = 36$ possible pairs.

Counting Cartesian product

Let $A_1, A_2, \ldots, A_m$ be finite sets.
We have that:

$$|A_1 \times A_2 \times \ldots \times A_n| = |A_1| \cdot |A_2| \cdot \ldots \cdot |A_n|$$

> *Proof* The task of choosing an element in the Cartesian product is done by choosing an element in A_1, an element in A_2 and so on until an element in A_m. This theorem then follows from the product rule.
>
> □

Example 1

We wonder how many different license plate can be made if each plate contains a sequence of 2 uppercase letters followed by 5 digits.
Since there are 26 different letters and 10 different digits, then this number is given by:

$$26^2 \cdot 10^5 = 67\,600\,000$$

Example 2

We wonder how many functions there are from a set with m elements to a set with n elements. For each of the m elements of the input, we have n possibilities in the other set. So, this number is given by:

$$n \cdot n \cdot \ldots \cdot n = n^m$$

Example 3

Let's say we now want to know how many one-to-one functions there are from a set with m elements to one with n elements. For the first element, we have n possibilities. Then, for the second one, we only have $n-1$ possibilities, since we want our function to be injective. And, so on until we have $n-(m-1)$ possibilities for the last element. Thus, this number is given by:

$$n(n-1)\cdot\ldots\cdot(n-m+1)=\frac{n!}{(n-m)!}$$

Example 4 We wonder what is the number of different subsets of a finite set S, $|P(S)|$.
Let T be any subset of S. We can make a sequence of 0s and 1s to say for each element of S if it is in T. Since there are 2 possibilities for each element (either it is in the set or not), we find by the product rule:

$$|P(S)|=2\cdot2\cdot\ldots\cdot2=2^n$$

Sum rule Let A and B be two tasks, with respectively n_1 and n_2 ways to do them.
If none of the set of n_1 ways is the same as any of the set of n_2 ways, then there are n_1+n_2 ways to do task A or B.

Example Le't say that a student can choose a semester project from one of three laboratories. The three laboratories offer 5, 3 and 7 possible projects, respectively. No project is offered by several laboratories.
We know by the sum rule that there are $5+3+7=15$ ways to choose a project.

Sum rule in terms of set Let A and B be sets. If A and B are disjoint sets, meaning $A\cap B=\emptyset$, then we know that:

$$|A\cup B|=|A|+|B|$$

This is the sum rule can be phrased in term of sets.
More generally, if we have n sets $A_1,\ldots,A_n$ such that $A_i\cap A_j=\emptyset$ for all i,j, then:

$$|A_1\cup\ldots\cup A_n|=|A_1|+\ldots+|A_n|$$

Example 1 Suppose variable names in a programming language can be either a single letter or a letter followed by a digit. We want to find the number of possible names. We have 26 possibilities for the single letter, and $26\cdot10$ for the letter followed by a digit. Thus, the number of variable names is given by:

$$26+26\cdot10=286$$

Example 2 Let's say that each user on a computer system has a password, which is 6 to 8 characters long, where each character is an uppercase letter or a digit. Each password must contains at least one digit.
By the sum rule, the number of passwords is given by:

$$P=P_6+P_7+P_8$$

For P_6, by the product rule, we have $(26+10)^6$ such passwords containing 6 upper case letters or digits. However, we to remove the passwords consisting of only letter. Thus, we get that:

$$P_6=(26+10)^6-26^6$$

Doing the same reasoning for P_7 and P_8, we get:

$$P=P_6+P_7+P_8=\left(36^6-26^6\right)+\left(36^7-26^7\right)+\left(36^8-26^8\right)=2\,483\,687\,808\,960$$

Subtraction rule If a task can be done either in one of n_1 ways or in one of n_2 ways, then the total number of ways to do the task in n_1+n_2 minus the number of ways to do the task that are common to the two different ways.
This is also known as the **principle of inclusion-exclusion**:

$$|A\cup B|=|A|+|B|-|A\cap B|$$

Example We wonder how many bit strings of length 8 either start with a 1 bit or end with the two bits 00.

Let A be the set of 8-bit bitstrings starting with 1, and B be the set of 8-bit bistrings ending with 00. We know that $|A| = 2^7$ since we fixed the first position and we have 7 other free positions. Using the same reasoning, we find that $|B| = 2^6$.
We still need to find the intersection. We know that $A \cap B = 2^5$ since there are 3 positions fixed (the bitstring begins with 1 and ends with 00) and 5 free positions.
So:

$$|A \cup B| = |A| + |B| - |A \cap B| = 2^7 + 2^6 - 2^5 = 160$$

12.2 Permutations and combinations

Definition of permutations A **permutation** of a set of n distinct objects is an *ordered* arrangement of these objects. An ordered arrangement or r elements of a set is called an ***r*-permutation**. We note this $P(n, r)$.

Example Let $S = \{1, 2, 3\}$. Then $3, 1, 2$ is an permutation of S. $3, 2$ is a 2-permutation of S.

Theorem (counting the number of permutations) Let n be a positive integer and r be an integer, where $1 \leq r \leq n$.
The number of r-permutations of a set with n distinct elements is given by:

$$P(n, r) = n(n-1) \cdot \ldots \cdot (n - r + 1)$$

Proof The first element can be chosen in n ways, the second in $n-1$ ways, and so on until the last element, which there are $m - (r - 1)$ ways to choose.

So, by the product rule:

$$P(n, r) = n(n-1) \ldots (n - r + 1)$$

Moreover, $P(n, 0) = 1$ since there is only one way to order zero elements.

Corollary We have that:

$$P(n, r) = n \cdot \ldots \cdot (n - r + 1) = \frac{n!}{(n-r)!}$$

Example We wonder how many ways are there to select a first-prize winner, a second prize winner and a third-prize winner from 100 different different people who have entered a contest.
This is number is given by:

$$P(100, 3) = \frac{100!}{(100-3)!} = \frac{100!}{97!} = 970\,200$$

Definition of combinations An ***r*-combination** of elements of a set (which contains n elements) is an *unordered* selection of r elements from the set. Thus, an r-combination is simply a subset of the set with r elements. This number is denoted:

$$C(n, r) \text{ or } \binom{n}{r}$$

We read the second one "n choose r".

Example Let S be the set $\{a, b, c, d\}$. Then, $\{a, c, d\}$ is a 3-combination from S. It is the same as $\{d, c, a\}$ since order does not matter.

Wednesday 24th November 2021 — **Lecture 20 : Repetitions and Pascal's triangle**

Theorem The number of r-combinations of a set with n elements, where $n \geq r \geq 0$, equals:

$$C(n,r) = \frac{n!}{(n-r)!r!}$$

> *Proof* By the product rule, we know that $P(n,r) = C(n,r) \cdot P(r,r)$: we choose the r elements, then we choose the order. Thus:
>
> $$C(n,r) = \frac{P(n,r)}{P(r,r)} = \frac{\frac{n!}{(n-r)!}}{\frac{r!}{(r-r)!}} = \frac{n!}{(n-r)!r!}$$
>
> since $0! = 1$.

Corollary Let n and r be nonnegative integers where $n \geq r \geq 0$. Then:

$$C(n, n-r) = C(n,r)$$

Example 1 We wonder how many hands of five cards can be dealt from a standard deck of 52 cards.
This is a basic combination:

$$C(52,5) = \frac{52!}{5!47!} = 2\,598\,960$$

We can see that it is the same number of hands of 47 cards:

$$C(52,47) = \frac{52!}{47!5!} = C(52,5)$$

And it makes sense since, when we are choosing 5 cards, we also choosing the 47 cards we are leaving behind.

Example 2 We wonder how many poker hands of five card with a full house (three of a kind, and a pair) can be dealt. An example of a full house is 2 aces and 3 kings.
We first take care of our three cards. To do that, we choose 1 face out of 13. Then for this, we choose 3 colours out of 4.
Second, we take care of our two cards left. We choose 1 face out of the 12 left (we cannot choose the same one as the one we took before), and we choose 2 colours out of the fours.
Thus, the total number is given by:

$$C(13,1) \cdot C(4,3) \cdot C(12,1) \cdot C(4,2) = 13 \cdot 4 \cdot 12 \cdot \frac{4}{2!2!} = 13 \cdot 4 \cdot 12 \cdot 6 = 3744$$

> *Vocabulary* The kind/face of a card is its value, its colour/suit is its "type" (there are four of them: club, diamond, heart and spade).

Definition An ***r*-permutation with repetitions** of a set of distinct objects is an ordered arrangement of r elements from the set, where elements can occur multiple times.

Theorem The number of r-permutations of a set of n objects with repetitions allowed is;

$$n^r$$

> *Proof* There are n ways to select an element of the set for each of the r positions, we can then use product rule to find n^r.

Example We wonder how many strings of length r can be formed from the upper-case letters of the English alphabet. This number is easily given by:

$$26^r$$

Definition An ***r*-combination** with repetition of elements of a set is an unordered selection of r elements from the set, where elements can occur multiple times.

Example Let's say we want to count the number of ways there are to select four pieces of apples, oranges, and pears if the order does not matter and the fruits are indistinguishable. The size of the set is $n = 3$, and we want to select $r = 4$ elements with repetitions. We can represent this as 3 compartiments: one for each fruit. We can do the following drawing where the points on the left of the two bars are apples, the ones that are between two bars are oranges, and the ones on the rightmost end are pears. On every line, there are fours dots, representing the fact that there are 4 fruits:

```
| | • • • •
| • | • • •
| • • | • •
| • • • | •
| • • • • |
• | | • • •
• | • | • •
• | • • | •
• | • • • |
• • | | • •
• • | • | •
• • | • • |
• • • | | •
• • • | • |
• • • • | |
```

We can see that we, in fact, we want to choose the $n - 1 = 2$ positions for the bar amongst the $r + n - 1 = 6$ possible. Another way of seeing that, is that we want to choose r positions for the fruits amongst the $r + n - 1 = 6$ possible. Thus, we have:

$$C(n + r - 1, n - 1) = C(n + r - 1, r) = C(6, 2) = \frac{6!}{4!2!} = 15$$

Theorem The number of r-combinations from a set with n elements when repetition of elements is allowed is:

$$C(n + r - 1, r) = C(n + r - 1, n - 1)$$

Proof The reasoning is similar as the one we did above.

Each r-combination of a set with n elements with repetition allowed can be represent by a list of $n - 1$ bars and r dots. The bars mark the n cells containing a dot for each time the i^{th} element of the set occurs in the combinaison.

The number of such lists is $C(n + r - 1, r)$: each list is a choice of the r positions to place the dots, from the total of $n + r - 1$ positions to place the dots and the bars.

This is also equal to $C(n + r - 1, n - 1)$, which is the number of ways to place the $n - 1$ bars.

□

Example 1 We wonder how many ways there are to select five bills of the following denominations: 1, 2, 5, 10, 20, 50 and 100.
We see that it is a r-combination with repetition, where $r = 5$ and $n = 7$. So, this number is given by:

$$C(n + r - 1, r) = C(11, 5) = \frac{11!}{5!6!} = 462$$

Example 2 We wonder how many solutions the equation $x_1 + x_2 + x_3 = 11$ has, where x_1, x_2 and x_3 are nonnegative integers.

Let's pick, x_1 is the number of apples, x_2 is the number of oranges and x_3 is the number of pears. Then, it is as if we want to pick 11 of those fruits, allowing repetitions. So, it is a r-combination where $n = 3$ and $r = 11$. Thus, this number is given by:

$$C(n+r-1,r) = C(13,11) = \frac{13!}{2!12!} = \frac{13 \cdot 12}{2} = 78$$

Example of permutations with indistinguishable objects

We wonder how many different strings can be made by reordering the letters of the word *SUCCESS*.

We see that, in this word, there are 3 S, 2 C, 1 U, 1 E, and a total of 7 characters. Finding the positions for the S is a combination: we choose 3 positions out of the 7 possible for them. Similarly, for the 2 C, there are C(4, 2) possibilities (there are already 3 positions occupied); for the 1 U there are $C(2,1)$; and for the 1 E there are $C(1,1)$.

Thus, all the number of possible ways is given by:

$$C(7,3) \cdot C(4,2) \cdot C(2,1) \cdot C(1,1) = \frac{7!}{3!4!} \cdot \frac{4!}{2!2!} \cdot \frac{2!}{1!1!} \cdot \frac{1!}{0!} = \frac{7!}{3!2!} = 420$$

Theorem

The number of different permutations of n objects, where there are n_1 indistinguishable objects type 1, n_2 of type 2, and so on until n_k indistinguishable objects of type k, is:

$$\frac{n!}{n_1!n_2! \cdot \ldots \cdot n_k!}$$

Proof

By the product rule, the total number of permutations is:

$$C(n,n_1)C(n-n_1,n_2) \cdot \ldots \cdot C(n-n_1-n_2-\ldots-n_{k-1},n_k)$$

Indeed, the n_1 objects of type 1 can be placed in the n positions in $C(n,n_1)$ ways, leaving $n-n_1$ positions. Then, the n_2 objects of type 2 can be placed in the $n-n_1$ positions in $C(n-n_1,n_2)$ ways, leaving $n-n_1-n_2$. We continue until we reach the n_k objects of type k.

So, we got:

$$\frac{n!}{n_1!(n-n_1)!} \cdot \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \cdot \ldots \cdot \frac{(n-n_1-\ldots-n_{k-1})!}{n_k!0!}$$

Which can be simplified to:

$$\frac{n!}{n_1!n_2! \cdot \ldots \cdot n_k!}$$

Distinction with permutations with repetitions

Permutations with repetitions and permutations with indistinguishable objects may seem confusing here is an example of both, to explain the distinction.

Let's pick $r = 3$, with $S = \{1,2\}$. The following examples are permutations with repetitions:

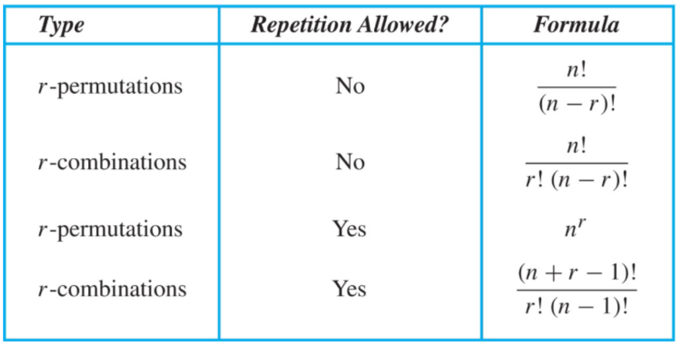
$$1,1,1 \quad 1,1,2 \quad 1,2,1 \quad 2,1,1 \quad 1,2,2 \quad 2,1,2 \quad 2,2,1 \quad 2,2,2$$

Let's pick $n_1 = 2$ and $n_2 = 1$ with (1, 1, 2). The following are permutation with indistinguishable objects:

$$1,1,2 \quad 1,2,1 \quad 2,1,1$$

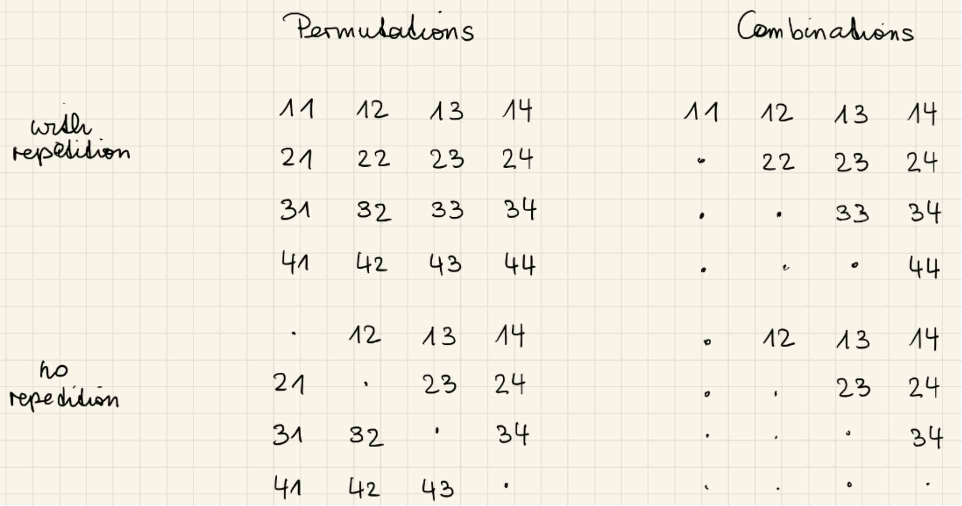
Summary

We can make the following summary:



Type	*Repetition Allowed?*	*Formula*
r-permutations	No	$\frac{n!}{(n-r)!}$
r-combinations	No	$\frac{n!}{r!\,(n-r)!}$
r-permutations	Yes	n^r
r-combinations	Yes	$\frac{(n+r-1)!}{r!\,(n-1)!}$

Moreover, if we take $n = 4$, $r = 2$ and $S = \{1, 2, 3, 4\}$, we have:



	Permutations				Combinations			
with repetition	11	12	13	14	11	12	13	14
	21	22	23	24	•	22	23	24
	31	32	33	34	•	•	33	34
	41	42	43	44	•	•	•	44
no repetition	•	12	13	14	•	12	13	14
	21	•	23	24	•	•	23	24
	31	32	•	34	•	•	•	34
	41	42	43	•	•	•	•	•

Poker dice We use five d6, which have the following faces: A, K, Q, J, 10, 9.
We ask ourselves the following questions:

1. What is the number of possible rolls?
2. What is the number of different outcomes (rolls, but unordered)?

We can do the following reasoning:

1. Each die gives us 6 possibilities, so this is a permutation with repetitions:

$$6^5 = 7776$$

2. This is a combination with repetitions:

$$C(n + r - 1, r) = C(10, 5) = 252$$

As expected, the same outcome can be realised by different rolls.

Pair Let's say we now want to know how many different outcomes there are for having one pair.

First, we choose 1 face for the pair. Second, we choose 3 faces for the other possibilities (we don't want a second pair). Both are combinations, so we find:

$$\binom{6}{1}\binom{5}{3} = 60$$

Now, let's say we wonder how many rolls realise each outcome. We have a permutation with repeated items, where $n = 5$, $n_1 = 2$ and $n_2 = n_3 = n_4 = 1$. So, for each of those 60 ways, there is the following number of different rolls that give them:

$$\frac{n!}{n_1!n_2!n_3!} = \frac{5!}{2!} = \frac{120}{2} = 60$$

Two pairs We can do the same computations for two pairs, we'll realise that here are the same number of different outcomes, 60, but less rolls that give them.

12.3 Binomial theorem

Example Let's expand the following polynomial:

$$(x+y)^3 = (x+y)(x+y)(x+y) = a_0x^3 + a_1x^2y + a_2xy^2 + a_3y^3$$

We wonder how we can get the coefficients without computing them. There is only one way of making the x^3, picking three times the x. For x^2y, we have to choose 2 factors from which we will take a x out of the 3. And we can continue. Thus, we have:

$$a_0 = \binom{3}{3}, \quad a_1 = \binom{3}{2}, \quad a_2 = \binom{3}{1}, \quad a_3 = \binom{3}{0}$$

Binomial theorem Let x and y be variables, and n a nonnegative integer. Then:

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j}y^j$$

Thus, the coefficients of the expansion of the powers of $x+y$ are related to the number of combinations.

Example Let's say we wonder what is the coefficient of $x^{12}y^{13}$ in the expansion of $(2x-3y)^{25}$. Using our theorem, we find:

$$(2x-3y)^{25} = ((2x)+(-3y))^{25} = \sum_{j=0}^{25} \binom{25}{j} (2x)^{25-j}(-3y)^j$$

We can then pick $j = 13$, and we find:

$$\binom{25}{13}(2x)^{12}(-3)^{13} = \frac{25!}{13!12!}2^{12}(-3)^{13}x^{12}y^{13}$$

Corollary Let $n \geq 0$. We have:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Intuition Every term is the number of different subsets of size k from a set of size n. Since we are adding them, it must give us the size of the power set, 2^n.

Proof

$$2^n = (1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k}1^k = \sum_{k=0}^{n} \binom{n}{k}$$

Pascal's identity If n and k are integers where $n \geq k \geq 0$, then:

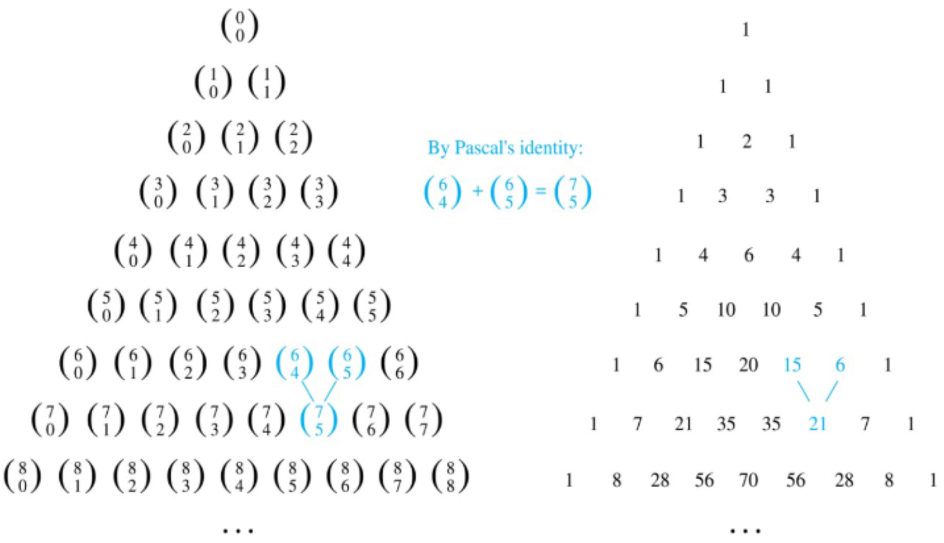
$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$$

Usefulness This is a recursive multiplication-free definition of the binomial coefficient.

Proof

$$\begin{aligned}
&\binom{n}{k-1} + \binom{n}{k} \\
&= \frac{n!}{(n-k+1)!(k-1)!} + \frac{n!}{(n-k)!n!} \\
&= \frac{n!k}{(n-k+1)!(k-1)!k} + \frac{n!(n-k+1)}{(n-k)!(n-k+1)k!} \\
&= \frac{n!k}{(n-k+1)!(k)!} + \frac{n!(n-k+1)}{(n-k+1)!k!} \\
&= \frac{n!(k+n-k+1)}{(n-k+1)!k!} \\
&= \frac{(n+1)!}{(n-k+1)!k!} \\
&= \binom{n+1}{k}
\end{aligned}$$

Pascal's triangle



Property The following identity holds:

$$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$$

Proof Let's consider the following function:

$$f(x) = (1+x)^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} x^k = \sum_{k=0}^{n} \binom{n}{k} x^k$$

Let's now consider this function's derivative:

$$f'(x) = \frac{d}{dx}(1+x)^n = n(1+x)^{n-1} \implies f'(1) = n2^{n-1}$$

Moreover:

$$f'(x) = \frac{d}{dx}\sum_{k=0}^{n} \binom{n}{k} x^k = \sum_{k=0}^{n} \binom{n}{k} k \cdot x^{k-1} \implies f'(1) = \sum_{k=0}^{n} k\binom{n}{k}$$

So:

$$f'(1) = f'(1) \implies \sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$$

□

Proof using combinatorics We choose a group of arbitrary size, and designate one of its element as a leader.

The first approach is to choose first the leader, we have n possibilities. Amongst the $n-1$ remaining, we choose whether they are in it or not, it is given by:

$$n2^{n-1}$$

The second approach is to choose a group of size k (it is given by $\binom{n}{k}$), and then choose the leader, for which we have k possibilities. This number is thus given by:

$$\sum_{k=1}^{n} k\binom{n}{k}$$

Since both approach must yield the same result, we have:

$$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$$

□

Tuesday 30th November 2021 — **Lecture 21 : Pigeons, pigeons, holes and recursive counting**

12.4 The pigeon-hole principle

The pigeon-hole principle If we have 13 pigeons and 12 holes, then necessarily we must have two 13 pigeons in the same hole. Let us generalise this concept.

Theorem Let $k \in \mathbb{N}_+$. If we have $k+1$ objects we want to put in k boxes, then at least one box contains two or more objects.

Proof Left as an exercise for the reader. Hint: use proof by contradiction.

Remark In French, we call this theorem *le principe des tiroirs et des chaussettes de Dirichlet.*

Corollary A function f from a set with $k+1$ elements to a set with k elements is not one-to-one.

Proof We apply the pigeon-hole principle. We take the elements of our domain as our pigeons and the elements from the codomain as our pigeon-holes.

Generalised version If N objects are placed into k boxes, then there is at least one box containing at least $\left\lceil \frac{N}{k} \right\rceil$ objects.

Proof Let's suppose for contradiction that none of the boxes has more than $\left\lceil \frac{N}{k} - 1 \right\rceil$ objects.

We notice that:

$$\left\lceil \frac{N}{k} \right\rceil < \frac{N}{k} + 1$$

Indeed, if $k \mid N$, then this is clearly true. Else, we have that $\left\lfloor \frac{N}{k} \right\rfloor < \frac{N}{k} < \left\lceil \frac{N}{k} \right\rceil$.

Thus, if we put in each box at most $\left\lceil \frac{N}{k} \right\rceil - 1$ objects, our total number of object cannot exceed the following number:

$$k\left(\left\lceil \frac{N}{k} \right\rceil - 1\right) < k\left(\frac{N}{k} - 1 + 1\right) = N$$

Since this means that there are less than N objects in total, it is a contradiction.

□

Example 1 Amongst 100 people, there are at least $\left\lceil \frac{100}{12} \right\rceil = 9$ who were born in the same month.

Example 2 Let's say we wonder how many cards must be selected from a standard deck of 52 cards to guarantee that at least three cards of the same suit are chosen.
We have N pigeons and 4 boxes. Thus, we want:

$$\left\lceil \frac{N}{4} \right\rceil \geq 3$$

We realise that $N = 12$ works:

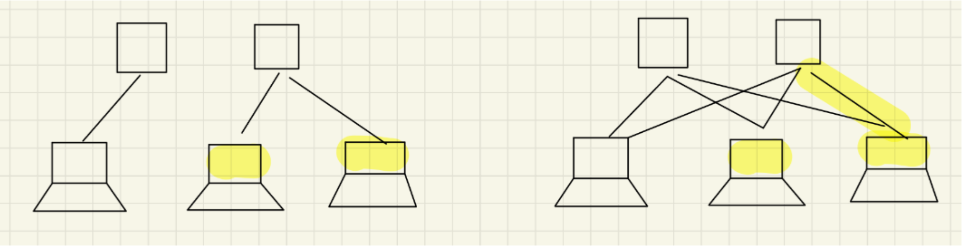
$$\left\lceil \frac{12}{4} \right\rceil = 3 \geq 3$$

Moreover, we see that $N = 11, 10, 9$ work as well:

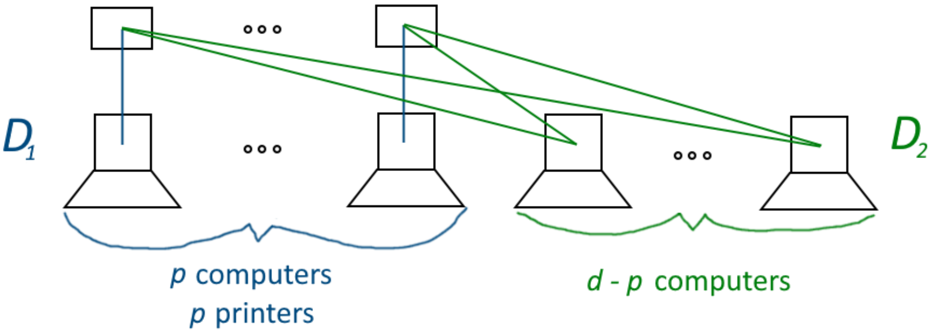
$$\left\lceil \frac{9}{4} \right\rceil = 3 \geq 3$$

Since $N = 8$ does not work, we have found that $N = 9$ is the smallest possible one.

Desktop Cabling Problem We have d computers and p printers, where $d > p$.
We wonder how many cables c are needed, connecting computers and printers such that whatever subset of p computers you select each has a separate cable to a printer. This allows p computer to print at the same time, whatever the computers we choose.



For example, the configuration on the left does not work, since the subset of the two rightmost computers is not such that each computer has its own printer. The one on the right works, but we could for example remove the cable in yellow.
Let us consider the following method:



In other words, the first p computers are directly linked to exactly one printer (different each), and the following $d-p$ computers are linked to all printers. We can show that this is a solution to the desktop cabling problem, and that it is the smallest number of cables that solves the problem.

Proof of solution

Let S be the set of selected computers, $|S| = p$. Let k, $0 \leq k \leq p$, be the number of computers from D_1 contained in S, i.e. $|S \cap D_1| = k$. They are connected using their unique cable.

This leaves $p-k$ unconnected printers, and S contains $p-k$ computers from D_2. Since there is a cable linking them all to all printer, we can distribute the printers. They can thus be connected.

Proof of optimality

Let's suppose for contradiction that this solution is not optimal, and thus that there exists a solution that works with a number of cable less than or equal to:

$$p + (d-p)\cdot p - 1$$

Let's use the pigeon-hole principle: our holes are our printers, and our pigeons are our computers. By the generalised version, we know that there is a printer P_r that is connected to the following number of computers:

$$\left\lfloor \frac{p + (d-p)p - 1}{p} \right\rfloor = \left\lfloor d - p + \left(1 - \frac{1}{p}\right)\right\rfloor = d - p$$

Let's now consider all computers that are not connected to P_r. There are at least p computers that are connected to the remaining $p-1$ printers, which is a contradiction (we cannot connect p computers to $p-1$ printers where each one would have its own printer, since this would be a subset which would break the problem).

□

12.5 Counting with recurrence relations

Definition

A **recurrence relation** for the sequence $\{a_n\}$ is an equation that expresses $\{a_n\}$ in terms of a finite number k of the preceding terms of the sequence, i.e:

$$a_n = f(n, a_{n-1}, a_{n-2}, \ldots, a_{n-k})$$

A sequence $\{a_n\}$ is called a **solution** of a recurrence relation if its terms satisfy the recurrence relation.
The **initial conditions** for a sequence specify the terms $a_0, a_1, \ldots, a_{k-1}$.

Method

1. Define a set P_n depending on a parameter n.
2. Describe P_n in terms of $P_{n-1}, \ldots, P_{n-k}$
3. Derive a recurrence relation for $|P_n|$.
4. Solve the recurrence relation.

Example 1

Let's say we want to count the number of bit strings of length n, which do not have two consecutive 0s.
First, let's define:

$$P_n = \left\{s \middle| |s| = n, \text{no consecutive 0s}\right\}, \quad n \geq 3$$

We can see easily that $|P_1| = 2$ (0 and 1) and $|P_2| = 3$ (01, 10 and 11).
Second, we see that for $s_n \in P_n$, then we have two cases: either s_n ends with a 1, then we can have $s_n = s_{n-1}1$, where $s_{n-1} \in P_{n-1}$; or s_n ends in a 0, then $s_{n-1}10$, where $s_{n-2} \in P_{n-2}$.

In other words, we can describe P_n the following way:

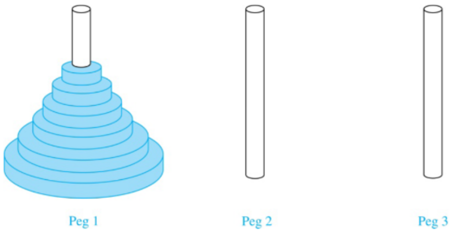
$$P_n = \left\{s_{n-1}1 \middle| s_{n-1} \in P_{n-1}\right\} \cup \left\{s_{n-2}10 \middle| s_{n-2} \in P_{n-2}\right\}$$

Third, we see that we get:

$$|P_n| = |P_{n-1}| + |P_{n-2}|$$

since the sets we take the union of are disjoint.

Example 2 In the Tower of Hanoi, we have three pegs and n disks. We are allowed to move the disks one at a time from one peg to another, as long as a larger disk is never placed on a smaller. The goal is to move all disks to the second peg (in order, with the largest on the bottom, since those are the rule). We wonder how many moves we need.



Let $\{H_n\}$ denote the number of moves needed to solve the Tower of Hanoi puzzle with n disks. The initial condition is $H_1 = 1$ since a single disk can be transferred from peg 1 to peg 2 in one move.
We can see that to move n objects from peg 1 to peg 2, we can move $n-1$ objects to peg 3, move the n^{th} to peg 2, and move back the $n-1$ to peg 2. Thus, we have the following recursive formula:

$$H_n = 2H_{n-1} + 1$$

Solving the recurrence Let's now say we want to solve this recurrence (in an informal way):

$$\begin{aligned} H_n &= 2H_{n-1} + 1 \\ &= 2^1(2H_{n-2} + 1) + 1 \\ &= 2^2 H_{n-2} + 2^1 + 1 \\ &= 2^2(2H_{n-3} + 1) + 2^1 + 1 \\ &= 2^3 H_{n-3} + 2^2 + 2^1 + 1 \\ &= \ldots \\ &= 2^{n-2}\big(2H_{n-(n-2)+1} + 1\big) + 2^{n-2} + 2^1 + 1 \\ &= 2^{n-1} \cdot 1 + 2^{n-2} + \ldots + 2^1 + 1 \\ &= 2^n - 1 \end{aligned}$$

Wednesday 1st December 2021 — **Lecture 22 : Closed forms of recurrence relations**

12.6 Solving recurrence relations

12.6.1 Linear homogeneous recurrence relations with constant coefficients

Definition

A **linear homogeneous recurrence relation of degree k with constant coefficients** is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

where $c_1, \ldots, c_k$ are real numbers and $c_k \neq 0$.

Observation By strong induction, a sequence satisfying such a recurrence relation is uniquely determined by the recurrence relation and the k initial conditions $a_0 = C_1, \ldots a_k = C_k$.

Terminology

- It is **linear** since the right-hand side is a sum of the previous term of the sequence each multiplied by a function of n.
- It is **homogeneous** because no terms occur that are not multiple of the a_js (there is no constant).
- It has **constant coefficients** $c_1, \ldots, c_k$.
- The **degree** is k because a_n is expressed in terms of the previous k terms of the sequence (the term "degree" will take all its sense with the characteristic equation).

Examples

- $P_n = 1.11 P_{n-1}$ is a linear homogeneous recurrence relation of degree one.
- $f_n = f_{n-1} + f_{n-2}$ is a linear homogeneous recurrence relation of degree two.
- $a_n = a_{n-1} + a_{n-2}^2$ is not linear.
- $H_n = 2H_{n-1} + 1$ is not homogeneous.
- $B_n = nB_{n-1}$ does not have constant coefficients.

Characteristic equation

Let's say we have the relation $a_n = c_1 a_{n_1} + \ldots + c_k a_{n-k}$. Then, we can guess that the solution is $a_n = r^n$, $r \in \mathbb{R}^*$, and see what happens:

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \ldots + c_k r^{n-k} \implies r^n - c_1 r^{n-1} - c_2 r^{n-2} - \ldots - c_k r^{n-k} = 0$$

Dividing by r^{n-k}, since it is non-zero (unless $a_n = 0$ for all n):

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \ldots c_k = 0$$

We call this the characteristic equation, and we can see that the sequence $\{r^n\}$ is a solution if and only if r is a solution to the characteristic equation.
The solutions to the characteristic equations are called the **characteristics roots** of the recurrence relation. The roots can be used to give a **closed formula** for the recurrence relation.

Theorem

Let c_1 and c_2 be real numbers. Moreover, let's suppose that $r^2 - c_1 r - c_2$ has two *distinct* roots r_1 and r_2.
The sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if:

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n, \quad n = 0, 1, 2, \ldots$$

where α_1 and α_2 are constants.

Proof We want to show that $\alpha_1 r_1^n + \alpha_2 r_2^n$ is a solution of $a_n = c_1 a_{n-1} + c_2 a_{n-2}$:

$$\alpha_1 r_1^n + \alpha_2 r_2^n = c_1\left(\alpha r_1^{n-1} + \alpha_2 r_2^{n-2}\right) + c_2\left(\alpha_2 r_1^{n-2} + \alpha_2 r_2^{n-2}\right)$$

Putting everything on the same side and grouping things, we get something which we want to prove that it is equal to 0:

$$\left(\alpha_1 r_1^n + c_1 \alpha_1 r_1^{n-1} - c_2 \alpha_1 r_1^{n-2}\right) + \left(\alpha_2 r_2^n - c_1 \alpha_2 r_2^{n-1} - c_2 a_2 r_2^{n-2}\right)$$

Thus, factoring:

$$\alpha_1 r_1^{n-2} \underbrace{\left(r_1^2 - c_1 r_1 - c_2\right)}_{=0} + \alpha_2 r_2^{n-2} \underbrace{\left(r_2^2 - c_1 r_2 - c_2\right)}_{=0} = 0$$

as required.

Note α_1 and α_2 depend on the initial conditions.

Example 1 Let's say we want to find the solution to the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$, with $a_0 = 2$ and $a_1 = 7$.
Let's get the characteristic equation by substituting $a_n = r^{n+2}$:

$$r^{n+2} = r^{n+1} + 2r^n \implies r^2 - r - 2 = 0 \implies (r+1)(r-2) = 0$$

Our roots are thus $r = -1$ and $r = 2$. Therefore, we know that solutions are of the form $a_n = \alpha_1 2^n + \alpha_2(-1)^n$. We know that $a_0 = 2$ and $a_1 = 7$, thus:

$$\begin{cases} 2 = a_0 = \alpha_1 + \alpha_2 & (12.2\text{a}) \\ 7 = a_1 = \alpha_1 \cdot 2 + \alpha_2(-1) = 2\alpha_1 - \alpha_2 & (12.2\text{b}) \end{cases}$$

This is a system of linear equations, which we can solve (using regular linear algebra methods), and which will give us:

$$\alpha_1 = 3, \quad \alpha_2 = -1$$

As a result, we get our closed form:

$$a_n = 3 \cdot 2^n - (-1)^n$$

Example 2 The sequence of Fibonacci numbers satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$, with the initial conditions $f_0 = 0$ and $f_1 = 1$.
This sequence has the following characteristic equation:

$$r^2 - r - 1 = 0 \implies r_1 = \frac{1+\sqrt{5}}{2} = \varphi, r_2 = \frac{1-\sqrt{5}}{2} = \overline{\varphi}$$

Substituting the initial conditions we get:

$$\begin{cases} 0 = f_0 = \alpha_1 + \alpha_2 \implies \alpha_1 = -\alpha_2 & (12.4\text{a}) \\ 1 = f_1 = \alpha_1\left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2\left(\frac{1-\sqrt{5}}{2}\right) & (12.4\text{b}) \end{cases}$$

Solving this system, we get:

$$\alpha_1 = \frac{1}{\sqrt{5}}, \quad \alpha_2 = -\frac{1}{\sqrt{5}}$$

Finally, we can put everything together:

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n = \frac{\varphi^n - \overline{\varphi}^n}{\sqrt{5}}$$

Theorem (degree 2 with repeated roots) Let c_1 and c_2 be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has one repeated root r_0.
The sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if:

$$a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$$

where α_1 and α_2 are constants.

Example Let's say we want to find the solution to the recurrence relation $a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1$ and $a_1 = 6$.
Its characteristic equation is given by:

$$r^2 - 6r + 9 = (r-3)^2 \implies r_0 = 3$$

Thus, our solution is of the form:

$$a_n = \alpha_1 3^n + \alpha_2 n 3^n$$

Using our initial conditions, we get:

$$\begin{cases} 1 = a_0 = \alpha_1 + \alpha_2 \cdot 0 = \alpha_1 & (12.6a) \\ 6 = a_1 = \alpha_1 \cdot 3 + \alpha_2 \cdot 1 \cdot 3^1 = 3\alpha_1 + 3\alpha_2 & (12.6b) \end{cases}$$

Solving it, we get:

$$\alpha_1 = 1, \quad \alpha_2 = 1$$

Thus:

$$a_n = 3^n + n3^n$$

Theorem Let $c_1, \ldots, c_k$ be real numbers. Let's suppose that the characteristic equation $r^k - c_1 r^{k-1} - \ldots - c_k = 0$ has k distinct roots $r_1, \ldots, r_k$.
Then, a sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}$ if and only if a_n is of the form:

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \ldots + \alpha_k r_k^n, \quad n = 0, 1, 2, \ldots$$

where $\alpha_1, \ldots, \alpha_k$ are constants.

Theorem Let $c_1, \ldots, c_k$ be real numbers. Let's suppose that the characteristic equation $r^k - c_1 r^{k-1} - \ldots - c_k = 0$ has t distinct roots $r_1, \ldots, r_t$, with multiplicities $m_1, \ldots, m_t$ respectively.
Then, a sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}$ if and only if a_n is of the form:

$$\begin{aligned} a_n = &\left(\alpha_{1,0} + a_{1,1} n + \ldots + \alpha_{1,m_1 - 1} n^{m_1 - 1}\right) r_1^n \\ &+ \left(\alpha_{2,0} + a_{2,1} n + \ldots + \alpha_{2,m_2 - 1} n^{m_2 - 1}\right) r_2^n \\ &+ \ldots \\ &+ \left(\alpha_{t,0} + a_{t,1} n + \ldots + \alpha_{t,m_t - 1} n^{m_t - 1}\right) r_t^n \end{aligned}$$

> *Personal note* This formula is not hard to learn when you understood it. We are basically generalising what we saw with degree 2.
>
> However, you can still note that using matrices simplifies the notation (this is more some "pretty" mathematical stuff than a useful

notation):

$$a_n = \begin{bmatrix} r_1^n \\ \vdots \\ r_t^n \end{bmatrix} \bullet \left(\begin{bmatrix} \alpha_1 & \ldots & \alpha_{1,m-1} \\ \vdots & & \vdots \\ \alpha_{t,0} & \ldots & \alpha_{t,m-1} \end{bmatrix} \begin{bmatrix} 1 \\ n \\ \vdots \\ n^{m-1} \end{bmatrix} \right)$$

where $m = \max(m_1, \ldots, m_t)$ and $\alpha_{p,q}$ is 0 if $q \geq m_p$ (we had to increase the number of coefficients because a matrix has the same number of coefficients on each row, but it's like setting all the unnecessary ones to 0).

Exemple Let's say we want to solve the recurrence relation $a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}$, with $a_0 = 1$, $a_1 = -2$ and $a_2 = -1$.
Its characteristic equation is given by:

$$r^3 + 3r^2 + 3r + 1 = 0 \iff (r+1)^3 = 0 \implies r_0 = -1$$

So, a_n is of the form:

$$\alpha_1(-1)^n + \alpha_2 n(-1)^n + \alpha_3 n^2(-1)^n$$

With our initial conditions, we get:

$$\begin{cases} 1 = a_0 = \alpha_1 \\ -2 = a_1 = \alpha_1(-1)^1 + \alpha_2 \cdot 1 \cdot (-1)^1 + \alpha_3 \cdot 1^2(-1) = -\alpha_1 - \alpha_2 - \alpha_3 \\ -1 = a_2 = \alpha_1(-1)^2 + \alpha_2 \cdot 2 \cdot (-1)^2 + \alpha_3 \cdot 2^2 \cdot (-1)^2 = \alpha_1 + 2\alpha_2 + 4\alpha_3^2 \end{cases}$$

(12.8a)

(12.8b)

(12.8c)

Solving the system, we get:

$$\alpha_1 = 1, \quad \alpha_2 = 3, \quad \alpha_3 = -2$$

So, the closed form of our sequence is given by:

$$a_n = (-1)^n + 3n(-1)^n - 2n^2(-1)^n$$

12.6.2 Generating functions

Definition The **generating function** for the infinite sequence $a_0, \ldots, a_k, \ldots$ of real numbers is the infinite series

$$G(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots = \sum_{k=0}^{\infty} a_k x^k$$

Examples The sequence $\{a_k\}$ with $a_k = k + 1$ has the generating function:

$$\sum_{k=0}^{\infty} (k+1)x^k$$

Similarly, if $a_k = 2^k$, then the generating function is given by:

$$\sum_{k=0}^{\infty} 2^k x^k$$

Remarkable generating functions We know the closed form of the following generating functions:

a_k	**Its generating function**
$C(n,k)$	$G(x) = (1+x)^n = \sum_{k=0}^{\infty} \binom{n}{k} x^k$
a^n	$G(x) = \frac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k$
$C(n+k-1,k)$	$G(x) = \frac{1}{(1-x)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$

We'll generalise binomial coefficients in the next lesson, we only need to know that $C(n,r) = 0$ when $r \geq n$.

Using generating functions Let's say we want to solve the recurrence relation $a_k = 3a_{k-1}$ with initial condition $a_0 = 2$, using generating functions.
We have:

$$G(x) = \sum_{k=0}^{\infty} a_k x^k = a_0 + \sum_{k=1}^{\infty} a_k x^k = a_0 + \sum_{k=1}^{\infty} 3a_{k-1}x^k = a_0 + 3x \sum_{k=1}^{\infty} a_{k-1}x^{k-1}$$

Changing the index, taking $k-1 \to k$, we get:

$$G(x) = a_0 + 3x \sum_{k=0}^{\infty} a_k x^k = a_0 + 3xG(x)$$

So, we can solve the equation:

$$G(x) = a_0 + 3xG(x) \iff G(x) = \frac{2}{1-3x} = 2 \cdot \sum_{k=0}^{\infty} 3^k x^k = \sum_{k=0}^{\infty} 2 \cdot 3^k x^k$$

We can see that, in the generating function, we have $a_k = 2 \cdot 3^k$. We could have solved this much simpler by "seeing it", or by using the method previously explained, but the point is that, now, we know how it works.

Solving Hanoi Tower When we wanted to solve our Hanoi Tower problem, we had found $H_n = 2H_{n-1} + 1$ and $H_0 = 0$. This is not homogeneous, so let's use generating functions:

$$G(x) = \sum_{k=0}^{\infty} H_k x^k = \underbrace{H_0}_{=0} + \sum_{k=1}^{\infty} H_k x^k = \sum_{k=1}^{\infty} (2H_{k-1} + 1)x^k$$

Let us split this function and then do the regular "factoring out-change of index trick":

$$G(x) = \sum_{k=1}^{\infty} 2H_{k-1}x^k + \sum_{k=1}^{\infty} x^k = 2x \sum_{k=1}^{\infty} H_{k-1}x^{k-1} + x \sum_{k=1}^{\infty} x^{k-1} = 2x \sum_{k=0}^{\infty} H_k x^k + x \sum_{k=0}^{\infty} x^k$$

We can now recognise our generating function, and one of the remarkable ones:

$$G(x) = 2xG(x) + x\frac{1}{1-x} \implies G(x) = \frac{x}{(1-2x)(1-x)}$$

Let us use partial fractions to split this fraction:

$$G(x) = \frac{x}{(1-2x)(1-x)} = \frac{a}{1-2x} + \frac{b}{1-x} = \frac{a(1-x) + b(1-2x)}{(1-2x)(1-x)}$$

So, we need to solve the following equation:

$$x = a(1-x) + b(1-2x) = a - ax + b - 2bx \iff (a+b) + x(-1-a-b) = 0$$

Thus we have the following system to solve:

$$\begin{cases} a+b=0 & (12.10a) \\ -1-a-b=0 & (12.10b) \end{cases}$$

Solving it, we find:

$$a=1, \quad b=-1$$

Coming back to our generating function, we have:

$$G(x)=\frac{a}{1-2x}+\frac{b}{1-x}=\frac{1}{1-2x}-\frac{1}{1-x}=\sum_{k=0}^{\infty}2^k x^k-\sum_{k=0}^{\infty}x^k=\sum_{k=0}^{\infty}(2^k-1)x^k$$

So, we have found $a_n=2^n-1$

General method

- Define the generating function $G(x)$.
- Use the recurrence relation to derive an alternative expression for $G(x)$ (using the "factoring out-change of index" trick).
- Solve the equation for $G(x)$.
- If it is a fraction of polynomial, use partial fractions to split it as a sum of terms of the form $\frac{c_i}{x-r_i}$ (it is always possible if the degree of the numerator is less than the one of the denominator).
- Determine the power expansion of $G(x)$, using remarkable generating functions or Taylor series (we have never used Taylor series and we will probably never use them, but I put them here because it would be a valid method if one forgot the remarkable generating functions).

. Tuesday 7th December 2021 — **Lecture 23 : Alienating sets, and *désolé pour le dérangement***

Fibonacci using generating function

As a recall, Fibonacci is given by:

$$f_n=f_{n-1}+f_{n-2}, \quad f_0=1, \quad f_1=1$$

Let's take a look at the generating function:

$$\begin{aligned} G(x) &= \sum_{k=0}^{\infty}f_k x^k \\ &= f_0+f_1x+\sum_{k=2}^{\infty}f_k x^k \\ &= f_0+f_1x+\sum_{k=2}^{\infty}(f_{k-1}+f_{k-2})x^k \\ &= f_0+f_1x+\sum_{k=2}^{\infty}f_{k-1}x^k+\sum_{k=2}^{\infty}f_{k-2}x^k \\ &= 1+x+x\sum_{k=2}^{\infty}f_{k-1}x^{k-1}+x^2\sum_{k=2}^{\infty}f_{k-2}x^{k-2} \\ &= 1+x+x\sum_{k=1}^{\infty}f_k x^k+x^2x^2\sum_{k=0}^{\infty}f_k x^k \\ &= 1+x+x(G(x)-f_0)+x^2G(x) \\ &= 1+x+x(G(x)-1)+x^2G(x) \end{aligned}$$

Solving this equation, we find:

$$G(x)=\frac{1}{1-x-x^2}=\frac{1}{-(x-r_1)(x-r_2)}, \quad r_1=\frac{-1-\sqrt{5}}{2}=-\varphi, r_2=\frac{\sqrt{5}-1}{2}=-\overline{\varphi}$$

Let's use partial fractions:

$$G(x) = \frac{1}{1-x-x^2} = \frac{a}{x-r_1} + \frac{b}{x-r_1} \implies a(x-r_2) + b(x-r_1) = 1$$

Solving this equation, we get $a = \frac{1}{\sqrt{5}}$ and $b = -\frac{1}{\sqrt{5}}$. So:

$$G(x) = \frac{1}{\sqrt{5}}\frac{1}{x-r_1} - \frac{1}{\sqrt{5}}\frac{1}{x-r_2} = \frac{-1}{\sqrt{5}r_1}\frac{1}{1-\frac{x}{r_1}} + \frac{1}{r_2\sqrt{5}}\frac{1}{1-\frac{x}{r_2}}$$

Using remarkable generating functions:

$$G(x) = -\frac{1}{r_1\sqrt{5}}\sum_{k=0}^{\infty}\frac{1}{r_1^k}x^k + \frac{1}{r_2\sqrt{5}}\sum_{k=0}^{\infty}\frac{1}{r_2^k}x^k$$

So, we get that:

$$f_n = \frac{\frac{-1}{r_1^{k+1}} + \frac{1}{r_2^{k+1}}}{\sqrt{5}} = \frac{\varphi^{k+1} - \overline{\varphi}^{k+1}}{\sqrt{5}}$$

12.7 More on generating functions

Combinations using generating functions

We want to distribute 2 cookies to 3 children such that no child receives more than 1 cookie, and we wonder in how many ways it can be done.

The first way of reasoning is that we want to choose two children to which we give the cookie, so we have $C(3, 2)$.

The second way of reasoning is considering children as the polynomial $(1 + x)$: the power of the x tells how many cookies this kid has. Considering all three children together, we have to expand the following polynomial:

$$(1+x)^3 = 1 + 3x + 3x^2 + x^3$$

Then, reading off the coefficient of x^2, we see that the result is 3. It basically states that x^2 can be formed as:

$$x \cdot x \cdot 1 = x^2, \quad x \cdot 1 \cdot x = x^2, \quad 1 \cdot x \cdot x = x^2$$

where each child is either represented by a 1 or a x: if they have a 1 they do not get a cookie, if they have a x they receive a cookie.

> *Personal note* Meanwhile, the poor Cookie Monster has zero cookie:
> `https://www.youtube.com/watch?v=1ZoIP4aoM0g`

Combinations

We want to find the number of k-combinations with n elements using generating functions.

The number of k-combinations is the coefficient of x^k in the following generating function:

$$f(x) = (1+x)^n = \sum_{k=0}^{n}\binom{n}{k}x^k$$

Thus, the number of k-combinations is $\binom{n}{k}$, as expected.

Definition (extended binomial coefficients)

Let u be a real number, and k be a nonnegative integer. The **extended binomial coefficient** is defined as:

$$\binom{u}{k} = \begin{cases} \dfrac{u(u-1)\cdots(u-k+1)}{k!}, & \text{if } k > 0 \\ 1, \quad \text{if } k = 0 \end{cases}$$

Example For, example:

$$\binom{-2}{3} = \frac{(-2)(-3)(-4)}{3!} = -4$$

$$\binom{1/2}{2} = \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)}{2!} = -\frac{1}{8}$$

Theorem (extended binomial theorem) Let x and u be real numbers, with $|x| < 1$. Then:

$$(1+x)^u = \sum_{k=0}^{\infty} \binom{u}{k} x^k$$

Observation Let's look at what happens when u is an integer, and k is greater than u. For example:

$$\binom{n}{n+1} = \frac{n(n-1)\cdots(n-n)}{(n+1)!} = 0$$

Thus, we are adding infinitely many zeroes when we only have integers. This is why, in the "regular" binomial theorem, we are only summing until n.

Combination with repetitions using generating functions We want to distribute 2 cookies to 3 children, and we wonder in how many ways it can be done.
Now, we can give each child 0, 1 or 2 cookies. This is a combination with repetition, so the number is given by:

$$C(3+2-1, 2) = C(4,2) = 6$$

Let's also consider this problem using generating functions. Again, let's consider that each child is a polynomial, $\left(1+x+x^2\right)$, where the power of x is the number of cookies we give to them. So, again, we need to read off the coefficient of x^2 in

$$\left(1+x+x^2\right)^3$$

Combinations with repetition We want to find the number of k-combinations of a set with n elements, where repetition is allowed, using generating functions.
The number of k-combinations is the coefficient of x^k in the generating function:

$$f(x) = \left(1+x+x^2+\ldots\right)^n$$

As long as $|x| < 1$, we get:

$$f(x) = \frac{1}{(1+(-x))^n} = (1-x)^{-n} = \sum_{k=0}^{\infty} \binom{-n}{k} (-x)^k = \sum_{k=0}^{\infty} \binom{-n}{k} (-1)^k (x)^k$$

So, the coefficient of x^k is $\binom{-n}{k}(-1)^k$, let's simplify it:

$$\begin{aligned}(-1)^k \binom{-n}{k} &= (-1)^k \frac{-n(-n-1)\cdots(-n-k+1)}{k!} \\ &= (-1)^k \frac{(-1)^k n(n+1)\cdots(n+k-1)}{k!} \\ &= (-1)^{2k} \binom{n+k-1}{k} \\ &= \binom{n+k-1}{k}\end{aligned}$$

as expected.

Example of generalisation 1 Let's now say that we want to distribute 8 cookies to 3 children, where each children receives 2, 3 or 4 cookies, and we wonder in how many ways it can be done.
We can translate it to a generating function, and we would only need to read off the coefficient of x^8 in:

$$\left(x^2+x^3+x^4\right)^3$$

Example of generalisation 2 Let's say we want to find the number of solutions of $e_1+e_2+e_3=17$, where e_1, e_2, e_3 are nonnegative integers with $2 \leq e_1 \leq 5$, $3 \leq e_2 \leq 6$ and $4 \leq e_3 \leq 7$. Translating this to a generating function would help us:

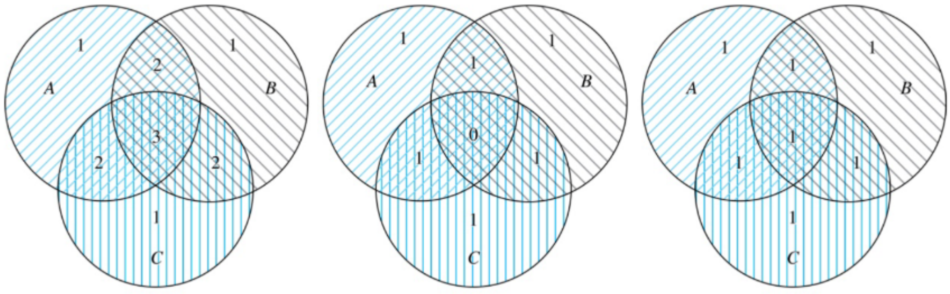
$$\left(x^2+x^3+x^4+x^5\right)\left(x^3+x^4+x^5+x^6\right)\left(x^4+x^5+x^6+x^7\right)$$

Note that this still is not very efficient since, computationally we are not far from counting every possible ways. We don't need to use generating functions; thinking a little bit about this problem is sufficient (we can for example see that each child must have at least 2 cookies, so we can diminish all our constraints by 2, and have $e_1+e_2+e_3=17-6=11$).

12.8 Principle of Inclusion-Exclusion

Three finite sets If we have three sets, we have:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$



(a) Count of elements by $|A|+|B|+|C|$

(b) Count of elements by $|A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|$

(c) Count of elements by $|A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|+|A\cap B\cap C|$

We could also derive it by expanding $|A \cup B \cup C|$ knowing that $|X \cup Y| = |X| + |Y| - |X \cap Y|$ (we have already proven that).

Example Let's say we wonder how many positive integers less that or equal to 1000 are divisible by 5, 7, or 11.
We can see that there are $\left\lfloor \frac{1000}{n} \right\rfloor$ integers less than or equal to 1000 divisible by n. Thus, the number we are looking for is given by:

$$\left\lfloor \frac{1000}{5} \right\rfloor + \left\lfloor \frac{1000}{7} \right\rfloor + \left\lfloor \frac{1000}{11} \right\rfloor - \left\lfloor \frac{1000}{5 \cdot 7} \right\rfloor - \left\lfloor \frac{1000}{5 \cdot 11} \right\rfloor - \left\lfloor \frac{1000}{7 \cdot 11} \right\rfloor + \left\lfloor \frac{1000}{5 \cdot 7 \cdot 11} \right\rfloor = 372$$

Theorem: Principle of Inclusion-Exclusion Let $A_1, \ldots, A_n$ be fine sets. Then $|A_1 \cup \ldots \cup A_n|$ is given by:

$$\sum_{1 \leq i \leq n} |A_i| - \sum_{i \leq i < k \leq n} |A_i \cap A_k| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \ldots + (-1)^{n+1} \left| \bigcap_{1 \leq i \leq n} A_i \right|$$

Proof Let's assume that an element a belongs to r sets, where $1 \leq r \leq n$.

When computing	a is counted _ times
$\sum \lvert A_i \rvert$	$r = C(r, 1)$
$\sum \lvert A_i \cap A_j \rvert$	$C(r, 2)$
$\sum \underbrace{\lvert A_i \cap \ldots \cap A_j \rvert}_{m \text{ sets}}$	$C(r, m)$

So, the total number of times a is counted is given by:

$$C(r, 1) - C(r, 2) + \ldots + (-1)^{i+1} C(r, r) = \sum_{i=1}^{r} (-1)^{i+1} C(r, i)$$

Moreover, we can notice that:

$$\begin{aligned} 0 &= (1 + (-1))^i \\ &= \sum_{i=0}^{r} (-1)^i C(r, r - i) \\ &= -\sum_{i=0}^{r} (-1)^{i+1} C(r, i) \\ &= -(-1)^{0+1} - \sum_{i=1}^{r} (-1)^{i+1} C(r, i) \\ &= 1 - \sum_{i=1}^{r} (-1)^{i+1} C(r, i) \end{aligned}$$

From this equation, we deduce that:

$$\sum_{i=1}^{r} (-1)^{i+1} C(r, i) = 1$$

which was the number we got, giving how many times a is counted. Each element is thus indeed counted exactly 1 time.

□

Definition: derangement A **derangement** is a permutation of objects that leaves no object in the original position.

Example The permutation of 21453 is a derangement of 12345. However, 21543 is not a derangement of 12345.

Personal note Derangements could for example be used to count the number of ways a Secret Santa can be organised. Indeed, we give everybody someone's name for whom they must buy a gift, but if someone must offer a gift to themself it is a bit sad.

Theorem The number of derangements of a set with n elements is given by:

$$D_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + (-1)^n \frac{1}{n!}\right)$$

Note This theorem is much less important than its proof. It is more important to be able to find this formula, rather than knowing the result by heart.

Proof

P_i are the permutations that fix element i, where $i = 1, \ldots, n$.
The set of permutations that fix at least one element is:

$$P_1 \cup P_2 \cup \ldots \cup P_n$$

From that, we get that the total number of derangements is given by:

$$D_n = n! - |P_1 \cup \ldots \cup P_n|$$

Using the principle of inclusion-exclusion:

$$|P_1 \cup \ldots \cup P_n| = \sum |P_i| - \sum_{i \neq j} |P_i \cap P_j| + \ldots + (-1)^{n+1} |P_1 \cap \ldots \cap P_n|$$

To compute $\sum |P_i|$, we see that we choose one element out of the n that we fix, and compute the permutations of the others:

$$C(n, 1)(n-1)!$$

Similarly, to compute $\sum_{i \neq j} P_i \cap P_j$, we can choose two elements that we fix, and then compute the permutations of the others:

$$C(n, 2)(n-2)!$$

The pattern appears, so we can compute what we were looking for:

$$\begin{aligned}
&|P_1 \cup \ldots \cup P_n| \\
&= \frac{n!}{(n-1)!1}(n-1)! - \ldots + (-1)^{n+1} \frac{n!}{(n-n)!n!}(n-n)! \\
&= n! - \frac{n!}{2} + \ldots + (-1)^{n+1} \frac{n!}{n!} \\
&= n! \left(\frac{1}{1!} - \frac{1}{2!} + \ldots + (-1)^{n+1} \frac{1}{n!} \right)
\end{aligned}$$

So, we do get that:

$$D_n = n! - |P_1 \cup \ldots \cup P_n| = n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \ldots + (-1)^n \frac{1}{n!} \right)$$

□

Example In the Hatcheck problem, a new employee checks the hats of n people at a restaurant, forgetting to put claim check numbers on the hat. When customers return for their hats, the checker gives them back hat chosen at random from the remaining hats. We wonder what is the chance that no one receives the correct hat.
This probability is given by:

$$\frac{D_n}{n!} = 1 - \frac{1}{1!} + \ldots + (-1)^n \frac{1}{n!}$$

Personal note It is interesting to see that, as $n \to \infty$, this probability tends towards e^{-1} ($\frac{D_n}{n!}$ is the power series of e^x evaluated at $x = -1$).

Wednesday 8th December 2021 — **Lecture 24 : Soon rich at a casino (by making one ofc)**

Chapter 13

Probabilities

13.1 Introduction and history

Short history The first treatise of probability was written by Cardano in the 16th century, because he was short on money and started gambling. Thus, probabilities began with gambling!

Then, in the 18th century, Bayes developed Bayes' theorem, which is the first concept of statistical data analysis. During the same century, Laplace developed the classical theory of probability, analysing games of chance.

In the 20th century, Kolmogorov developed the modern axiomatic of probability theory, yielding probabilities over real numbers. This is much more complex than what we will see.

Definitions An **experiment** is a procedure that yields one **outcome** out of a given set. The **sample space**, S, of the experiment is the set of possible outcomes. An **event** E is a subset of the sample space. An event **occurs** if the outcome belongs to the sample space.

> *Example* Rolling a dice is an experiment, its sample space is $S = \{1, 2, 3, 4, 5, 6\}$, an event could be $E = \{5, 6\}$ (rolling a 5 or a 6), and, for instance, this even occurs when rolling a 6.

Definition of probability If S is a finite sample space of equally likely outcomes, and E is an event (so a subset of S), then the **probability** of E is:

$$p(E) = \frac{|E|}{|S|}$$

> *Observation* We notice that, for every event E, we have:
>
> $$0 \leq p(E) \leq 1$$
>
> Indeed, $E \subset S$, so $0 \leq |E| \leq |S|$. By convention, no probability is negative or strictly greater than 1.

> *Note* This definition is known as the "classical definition of probability", or Laplace's definition. Indeed, in Laplace's *Théorie Analytique des Probabilités*, he writes:
>
> "`La théorie des probabilités consiste à réduire tous les événements qui peuvent avoir lieu dans une circonstance donnée, à un certain nombre de cas également possibles, c'est-à-dire tels que nous soyons également indécis sur leur existence,`

> et à déterminer parmi ces cas, le nombre de ceux qui sont favorables à l'événement dont on cherche la probabilité. Le rapport de ce nombre à celui de cas possibles, est la mesure de cette probabilité qui n'est donc qu'une fraction dont le numérateur est le nombre des cas favorables, et dont le dénominateur est celui de tous les cas possibles."
>
> Wikipedia (accessed on 8$^{\text{th}}$ December 2021) writes it more concisely, in English:
>
> "The probability of an event is the ratio of the number of cases favorable to it, to the number of all cases possible when nothing leads us to expect that any one of these cases should occur more than any other, which renders them, for us, equally possible."

> *Fallacies* Note that we always have to be careful when applying probabilities. For example, the prosecutor's fallacy (on which I will come back in the next lesson (aha it's a cliffhanger)) has already led to people getting convicted for crimes they had not done.

Example We wonder what is the probability that, when rolling two (fair) dice, their sum gives 7.

By basic combinatorics, both dice can give 6 values, so $|S| = 6 \cdot 6 = 36$. For E, we can list the possibilities:

$$E = \{(1,6),(2,5),(3,4),(4,3),(5,2),(6,1)\} \implies |E| = 6$$

So, we can compute the probability:

$$p(E) = \frac{6}{36} = \frac{1}{6}$$

Now, let's compute the probability to have a 6 in one of those dice (that at least one die rolls a 6). Again, let's list the elements of E_2:

$$E_2 = \{(1,6),(2,6),(3,6),(4,6),(5,6),(6,6),(6,1),(6,2),(6,3),(6,4),(6,5)\}$$

So, $|E_2| = 11$. We can now compute $p(E_2)$:

$$p(E_2) = \frac{11}{36} \approx 0.31$$

> *Terminology* Note that, in English, we say "one die" (singular), and "multiple dice" (plural).
>
> This is not absolute, it is the subject of controverses amongst English speakers: some say that we should use the word "dice" in singular. In any case, the word "dices" does not exist as a noun. Anyhow, we will use the convention explained hereinabove in this course.

Example 2 For the EuroMillions, we choose a set of 5 numbers out of 50 and 2 numbers out of 12. We wonder what is the probability that a person picks the correct 7 numbers.

The number of ways to choose 5 numbers out of 50 and to choose 2 number out of 12 is, by the product rule:

$$C(50,5)C(12,2) = 139\,838\,160$$

Since there is only one exact combination that gives the win, the size of the event space is 1. Thus, the probability of winning is given by:

$$\frac{1}{139\,838\,160} \approx 0.000000715\,\% = 7.15 \cdot 10^{-9}$$

Example 3 Let's say we wonder what is the probability that the numbers $11, 4, 17, 39, 23$ are drawn in that order from a bin with 50 balls labelled with the numbers from 1 to 50. If the ball selected is not returned to the bin, then the size of the sample space is given by a permutation; the size of the event is 1 since there is exactly winning combination. So, the probability is given by:

$$\frac{1}{50 \cdot 49 \cdot 48 \cdot 47 \cdot 46} = \frac{1}{254\,251\,200}$$

Now, if the ball is returned to the bin before the next ball is selected, we have a permutation with repetitions, so the probability is given by:

$$\frac{1}{50^5} = \frac{1}{312\,500\,000}$$

13.2 Properties of probabilities

Theorem: complements of events Let E be an event in a sample space S. The probability of the event $\overline{E} = S \setminus E$, the **complementary events** of E, is given by:

$$p\left(\overline{E}\right) = 1 - p(E)$$

Proof Using set theory, we get:

$$p\left(\overline{E}\right) = \frac{\left|\overline{E}\right|}{|S|} = \frac{|S \setminus E|}{|S|} = \frac{|S| - |E|}{|S|} = 1 - p(E)$$

□

Example The probability not to roll a 6 with 2 dice is:

$$1 - \frac{11}{36} = \frac{25}{36} \approx 70\,\%$$

Example A sequence of 10 bits is chosen randomly. We wonder what is the probability that at least one of these bits is 0.
We notice that the total number of strings is given by:

$$|S| = 2^{10} = 1024$$

The probability that all bits are 1 is given by $\frac{1}{1024}$, thus the probability that there is at least one 0 is given by:

$$1 - \frac{1}{1024} = \frac{1023}{1024}$$

Theorem: probability of unions Let E_1 and E_2 be events in a sample space S. Then:

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$$

Proof The proof follows from the inclusion-exclusion formula.

Example 1 Let the sample space be positive integers not exceeding 100. We wonder what is the probability that a positive integer selected at random from this sample space is divisible by either 2 or 5.
Let E_1 be the event that the number is divisible by 2, and E_2 that the number is divisible by 5. This yields that $E_1 \cup E_2$ is the even that the integer is divisible by 2 or 5, and $E_1 \cap E_2$ that the integer is divisible by 2 and 5.

So, we can compute our probability:

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2) = \frac{50}{100} + \frac{20}{100} - \frac{10}{100} = \frac{3}{5}$$

Example 2 Let's get back a previous example: we roll two dice and we want to know the probability to roll a 6.
Let's pick E_1 to be the event that the first die is 6, and E_2 that the second die is 6.
Thus, we have:

$$p(E_1 \cup E_2) = \frac{1}{6} + \frac{1}{6} - \frac{1}{36} = \frac{11}{36}$$

13.3 Probabilities without equally likely outcomes

Limitation of Laplace's definition We assumed that all outcomes are equally likely, but this is not the case.
For example, let's compute the sum of rolling two dice. The set of possible outcomes is

$$S = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

However, drawing the following table, we can see that they are not all equally likely:

$+$	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Indeed, for example the sum gives 7 six times more often than 2.

Definition: Probability distribution Let S be a sample space of an experiment with a finite or countable number of outcomes. We assign a probability $p(s)$ to each outcome s, so that:

1. $0 \leq p(s) \leq 1$, for each $s \in S$
2. $\displaystyle\sum_{s \in S} p(s) = 1$

The function p from the set of all outcomes of the sample space S is called a **probability distribution**.

Example 1 We roll two dice, and compute their sum. The set of possible outcomes is:

$$S = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

Looking at the table drawn above, there each diagonal only has one number, so we get that the probability distribution is given by:

$$p(7 - i) = \frac{6 - |i|}{36}, \quad -5 \leq i \leq 5$$

So, more explicitly:

$$p(i) = \frac{6 - |7 - i|}{36}, \quad 2 \leq i \leq 12$$

Example 2 Let's take a biased coin such that heads (H) come up twice as often as tails (T).
We wonder what probabilities we should assign to the outcomes H and T.
This gives us a first equation:

$$P(H) = 2P(T)$$

Moreover, since probabilities must add up to one (it is in the definition of probability distributions):

$$P(H) + P(T) = 1$$

This is a system of 2 equations with 2 unknowns. Solving it, we get:

$$P(H) = \frac{2}{3}, \quad P(T) = \frac{1}{3}$$

Definition: Uniform distribution Let S be a set with n elements. The **uniform distribution** assigns the probability $\frac{1}{n}$ to each elements of S, i.e:

$$p(s) = \frac{1}{n}, \quad \forall s \in S$$

Example The distribution in a fair coin-flipping experiement is uniform:

$$p(H) = P(T) = \frac{1}{2}$$

Note Laplace's definition supposes we only work with uniform distributions.

Definition: Probability of an event The **probability** of the event E is the sum of the probabilities of the outcomes in E. In other words:

$$p(E) = \sum_{s \in E} p(s)$$

Example Let's say that we have a biased die so that 3 appears twice as often as each other number, but that the other five outcomes are equally likely. We wonder what is the probability that an odd numbers appears when we roll this die.
The informations above give us the following equation:

$$p(3) = 2P, \quad p(s) = P \ \forall s \in \{1, 2, 4, 5, 6\}$$

Moreover, since probabilities have to add up to 1:

$$\sum_{s \in S} p(s) = 5P + p(3) = 7P = 1 \implies P = \frac{1}{7}$$

So, the probability of having an odd number is given by:

$$\sum_{s \in \{1,3,5\}} p(s) = p(1) + p(3) + p(5) = \frac{4}{7}$$

which is higher than with a fair die.

Complements and unions The two following properties still hold with our new definition:

$$p\left(\overline{E}\right) = 1 - p(E), \quad p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$$

Proof of the first equality This is a direct proof:

$$p(\overline{E}) = \sum_{s\in\overline{E}} p(s) = \sum_{s\in S\setminus E} p(s) = \sum_{s\in S} p(s) - \sum_{s\in E} p(s) = 1 - p(E)$$

□

13.4 Conditional probability

Introduction Often, probabilities exist in some context, or when a certain condition is satisfied. For example, we may wonder what is the chance to test positive on Covid-19. This is probably a different probability than the chance to test positive on Covid if one feel sick.

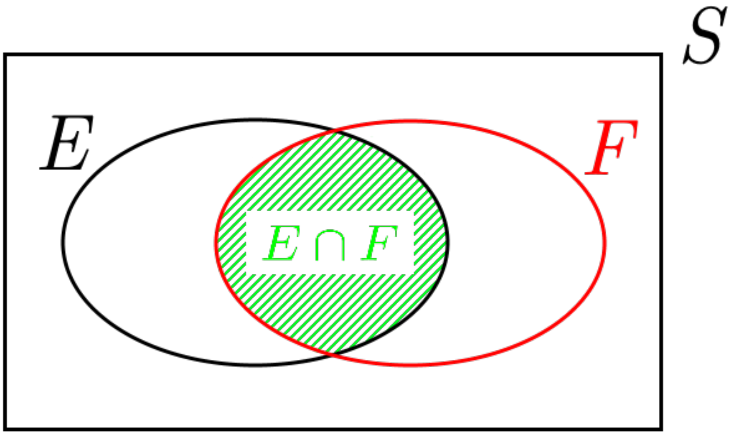
Moreover, the chance of having a head if the last 5 coin tosses were tails is the same as having a head without any context; but it would be nice to have a way to prove this.

Definition: conditional probability Let E and F be events, with $p(F) \neq 0$. The **conditional probability** of E given F, denoted by $p(E|F)$, is defined as:

$$p(E|F) = \frac{p(E\cap F)}{p(F)}$$

It can be interpreted as the probability that E occurs, give the fact (or knowing) that F occurs.

Intuition It is like if we had a new sample space: F.



Example 1 We roll a dice, and we wonder what is the probability that the outcome is even. Without any additional knowledge, this is $\frac{3}{6} = \frac{1}{2}$.

However, if we know that the roll gave a number less than or equal to 3, then we need to compute conditional probability. Let E be the event that the outcome is even, and F be that the die gives a value less than or equal to 3. Both $p(E)$ and $p(F)$ are $\frac{1}{2}$. Finally, $E\cap F$ is the event that the outcome is even and it is less than or equal to three, so $p(E\cap F) = \frac{1}{6}$ (the only possibility is that the die gives 2). Thus:

$$p(E|F) = \frac{p(E\cap F)}{p(F)} = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3}$$

Example 2 We toss a coin 6 times. We wonder what is the probability that the last toss is heads. If we don't know anything more, then the probability is $\frac{1}{2}$.

Let's now analyse the case in which the first five tosses are tails. Let E be the event that the last toss is heads ($p(E) = \frac{1}{2}$) and F be that the first 5 tosses are tail ($p(F) = \frac{1}{2^5}$). We see that $E \cap F$ is the event of having 5 tosses followed by a head and its probability is given by $p(E \cap F) = \frac{1}{2^6}$ (this event is an exact sequence, so this is the reciprocal of the sample space size). So, we can compute the probability:

$$p(E|F) = \frac{p(E \cap F)}{p(F)} = \frac{\frac{1}{2^6}}{\frac{1}{2^5}} = \frac{1}{2}$$

This is often known as the gambler fallacy, or Monte Carlo fallacy.

Intuition This makes sense, since the coin has "no way to know" what the previous events were. If there was a higher probability of getting heads after having tails, then what would happen if we toss 5 heads, wait a day and toss the coin again? What if it were somebody else who tossed the coin? Those questions reveal that there must be a problem, and thus that the probability is indeed $\frac{1}{2}$.

Definition: Independence The events E and F are **independent** if and only if:

$$p(E \cap F) = p(E)p(F)$$

Example The result of tossing coins (getting tail or head) are independent events.

Theorem: Independence If E and F are independent, then:

$$p(E|F) = p(E)$$

Proof This is a direct proof:

$$p(E|F) = \frac{p(E \cap F)}{p(F)} = \frac{p(E)p(F)}{p(F)} = p(E)$$

□

Example 1 Let's assume that each of the four ways a family can have two children $\{BB, GG, BG, GB\}$ (where B means boy, G means girl, in a world where there are exactly two sexes) is equally likely.
Let $E = \{BG, GB\}$ be the event that a family with tow children has both girls and boys, and $F = \{GG, BG, GB\}$ that a family with two children has at most one boy. We wonder if E and F are independent.
We can deduce that $p(E) = \frac{1}{2}$ and $p(F) = \frac{3}{4}$. Moreover, $E \cap F = \{BG, GB\}$, so $p(E \cap F) = \frac{1}{2}$. Thus:

$$p(E)p(F) = \frac{3}{8} \neq \frac{1}{2} = p(E \cap F)$$

So, those two events are not independent.

Example 2 Let's ask ourselves the same question, but with a family who has three children:

$$S = \{BBB, BBG, BGB, GBB, GGB, GBG, BGG, GGG\} \implies |S| = 8$$

We have:

$$E = \{BBG, GGB, BGB, BGG, GBB, GBG\} \implies p(E) = \frac{6}{8} = \frac{3}{4}$$

$$F = \{GGB, BGG, GBG, GGG\} \implies p(F) = \frac{4}{8} = \frac{1}{2}$$

Now, we can look at their intersection:

$$E \cap F = \{GGB, BGG, GBG\} \implies p(E \cap F) = \frac{3}{8}$$

Finally, we can see that those events are independent:

$$p(E)p(F) = \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8} = p(E \cap F)$$

Increasing the size of the sample space can change the dependence of events. This means that we must not trust our intuition.

Definition: pairwise independence The events $E_1, \ldots, E_n$ are **pairwise independent** if and only if:

$$p(E_i \cap E_j) = p(E_i)p(E_j), \quad \forall i, j, i < j \leq n$$

Mutual independence The events $E_1, \ldots, E_n$ are **mutually independent** if and only if:

$$p(E_1 \cap \ldots \cap E_n) = p(E_1) \cdot p(E_2) \cdots p(E_n)$$

Theorem: Independence If events are mutually independent, then they are pairwise independent.

Reciprocal The reciprocal is wrong. For example, let's say we are tossing a fair coin twice.

Let's take E_1 be the event that the first toss is 1, E_2 be that the second toss is 1 and E_3 be that the two tosses have a different result. We have:

$$p(E_1) = \frac{1}{2}, \quad p(E_2) = \frac{1}{2}, \quad p(E_3) = \frac{1}{2}$$

E_1, E_2, E_3 are pairwise independent:

$$p(E_1 \cap E_2) = \frac{1}{4} = p(E_1)p(E_2)$$

$$p(E_1 \cap E_3) = \frac{1}{4} = p(E_1)p(E_3)$$

$$p(E_2 \cap E_3) = \frac{1}{4} = p(E_2)p(E_3)$$

However, they are not mutually exclusive:

$$p(E_1 \cap E_2 \cap E_3) = 0 \neq \frac{1}{8} = p(E_1)p(E_2)p(E_3)$$

Analysis of side information We roll a pair of dice remotely (we don't see the result). Then, an observer tells us some property of the dice. Depending on what he says, we decide whether we bet that the sum is 7.

Without side information, we know that the probability is $\frac{1}{6}$.

Case 1 Let's say that the observer tells us that "the sum is 7". Then we know that we can bet, since we have a 100% chance of winning.

Case 2 Let's now say that the observer is less generous and says "the roll contains a 6".

Let's use conditional probability to study the result. Let S_7 be the event that the sum is 7, and R_6 be that the roll contains a 6. We notice that $|S_7 \cap R_6|$ is 2, since there are only two ways of getting a sum of 7 with a 6 ($(1, 6), (6, 1)$). So, the probability is given by:

$$S(S_7 | R_6) = \frac{p(S_7 \cap R_6)}{p(R_6)} = \frac{\frac{2}{36}}{\frac{11}{36}} = \frac{2}{11} > \frac{1}{6}$$

So, this would be a good idea to bet. However, this is a bit counter-intuitive, since we would get the exact same result if the observer told us that the roll contains any other number. Why does this give us more information, this seems very counter-intuitive? Let's consider the following case before answering this question.

Case 3 Let's now say that the observer tells us "the dice are different". The probability is given by:

$$p(S_7|R_{diff}) = \frac{p(S_7 \cap R_{diff})}{p(R_{diff})} = \frac{\frac{1}{6}}{\frac{5}{6}} = \frac{1}{5} > \frac{1}{6}$$

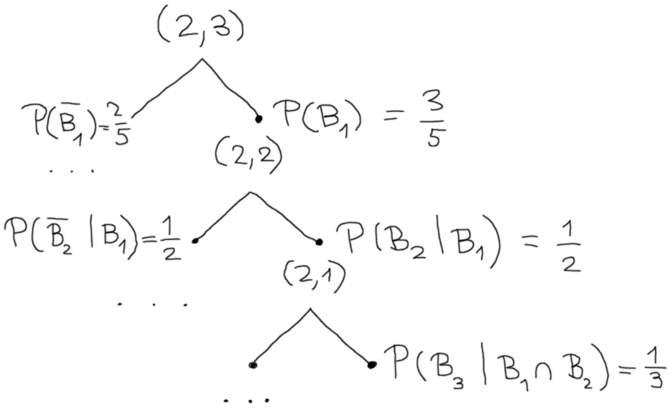
This is increasing our chances, and it makes sense since we are eliminating "bad" cases (there is no way of reaching 7 with dice that give the same result). This makes us understand more the second case: by only sticking to one number, we are removing some bad results.

Tuesday 14th December 2021 — **Lecture 25 : Bayes-ed**

Probability trees We can use trees to analyse complex events which use conditional probabilities.

For example, let's say we have a bag with 2 white marbles and 3 black ones. We wonder what is the probability to draw 3 black marbles in a row, if we take 3 marbles from the bag.

Let's define B_i with $i = 1, 2, 3$ to be the event that the i^{th} marble is black. We can draw the following tree:



Now, we can see that:

$$p(E|F) = \frac{p(E \cap F)}{p(F)} \implies p(E \cap F) = p(E|F)p(F)$$

So:

$$p(B_1 \cap B_2 \cap B_3) = p(B_3|B_1 \cap B_2)p(B_1 \cap B_2) = p(B_3|B_1 \cap B_2)p(B_2|B_1)p(B_1)$$

We thus get that:

$$p(B_1 \cap B_2 \cap B_3) = \frac{1}{3}\frac{1}{2}\frac{3}{5} = \frac{1}{10}$$

Note that the following notation is often used:

$$p(B_3|B_1 \cap B_2) = p(B_3|B_1, B_2)$$

13.5 Bernoulli trials

Example Let's say we have a biased coin that gives head 2 times out of 3. We wonder what is the probability that exactly four heads occur when the coin is flipped seven times. There are $C(7,4)$ possibilities to have 4 heads. Independently of the order, the probability to have 4 heads is given by $\left(\frac{2}{3}\right)^4\left(\frac{1}{3}\right)^3$. So, we get our probability:

$$C(7,4)\left(\frac{2}{3}\right)^4\left(\frac{1}{3}\right)^3 = \frac{560}{2187} \approx 25.61\,\%$$

Definition: Bernoulli trials Let's say we have an experiment that can have only two possible outcomes, one is called a **success** and the other a **failure**. If p is the probability of success and q the probability of failure, then we have $p+q=1$, since those are our two only possible outcomes. Each performance of the experiment is called a **Bernoulli trial**.

A frequent question over Bernoulli trials is to determine the probability of k successes when an experiment consists of n mutually independent Bernoulli trials.

Theorem The probability of exactly k successes in n independent Bernoulli trials, with probability of success p and probability of failure $q = 1-p$ is:

$$C(n,k)p^k q^{n-k}$$

Binomial Distribution We denote $b(k:b,p)$ the probability of k successes in n independent Bernoulli trials with probability of success p. Viewed as a function of k, $b(k:n,p)$ is the **binomial distribution**.

By the theorem above:

$$b(k:n,p) = C(n,k)p^k q^{n-k}$$

Example We want to analyse our exams. We wonder what is the probability of guessing at least 3 questions correctly out of 6, with 4 possible answers.

This is a binomial distribution with $p = \frac{1}{4}$ and $n = 6$. The probability of guessing 0, 1 and 2 questions is given by:

$$p(E_0) = \binom{6}{0}\left(\frac{1}{4}\right)^0\left(\frac{3}{4}\right)^6 = \frac{3^6}{4^6}$$

$$p(E_1) = \binom{6}{1}\left(\frac{1}{4}\right)^1\left(\frac{3}{4}\right)^5 = \frac{6\cdot 3^5}{4^6}$$

$$p(E_2) = \binom{6}{2}\left(\frac{1}{4}\right)^2\left(\frac{3}{4}\right)^4 = \frac{6\cdot 5\cdot 3^4}{2\cdot 4^6}$$

So, the probability not to answer at least 3 questions is given by their sum, 0.68, since they are mutually exclusive. We thus know that the probability to pass by chance is approximately $1-0.68 = 0.32$ (so do not solve the exam by only guessing).

13.6 Bayes' Theorem

Motivation This theorem allows us to for example answer the following question: "given that someone tests positive for having the Corona Virus, what is the probability that they actually do have this virus?". Similarly, we may wonder what is the probability that someone actually has the virus if they tests negative.

Bayes' Theorem Let E and F be events from a sample space S such that $p(E) \neq 0$ and $p(F) \neq 0$. Then:

$$p(F|E) = \frac{p(E|F)p(F)}{p(E)} = \frac{p(E|F)p(F)}{p(E|F)p(F) + p(E|\overline{F})p(\overline{F})}$$

Intuition

Basically, this theorem allows us to switch what is given in the conditional probability. For example, it allows us to go from "the probability to test positive knowing that we have the virus" to "the probability to have the virus knowing that we testes positive".

We have to be careful, since our intuition often wants to make us think that $p(E|F) = p(F|E)$, which is generally false. This fallacy even has a name: it is called *the prosecutor's fallacy*. Note that such fallacy has already led to people getting convicted for a crime they had not done. Here is a video from *Vsauce2* speaking about such a crime:

`https://www.youtube.com/watch?v=mTNlVAz2fdA`

Example

Let E be the event of testing positive on Corona, and F be the event of having Corona.

The **prior probability**, $p(F)$, is the probability that someone has Covid (so, the percentage of the population that has the virus). The **evidence**, $p(E)$, is the probability of testing positive, this comes from statistical studies. The **likelihood**, $p(E|F)$ is the probability of testing positive when someone has Corona, this also comes from statistical studies. Finally, the **posterior probability**, $p(F|E)$ is what we are looking for.

Proof

By definition of conditional probability:

$$p(F|E) = \frac{p(E \cap F)}{p(E)} = \frac{p(E \cap F)}{p(F)} \cdot \frac{p(F)}{p(E)} = \frac{p(E|F)p(F)}{p(E)}$$

For the full version of this theorem, we notice the following equality:

$$p(E) = p\big((E \cap F) \cup \big(E \cap \overline{F}\big)\big) = p(E \cap F) + p\big(E \cap \overline{F}\big)$$

since $E \cap F$ and $E \cap \overline{F}$ are disjoint event.

Now, using the equality we saw earlier in this lesson:

$$p(E) = p(E|F)p(F) + p\big(E|\overline{F}\big)p\big(\overline{F}\big)$$

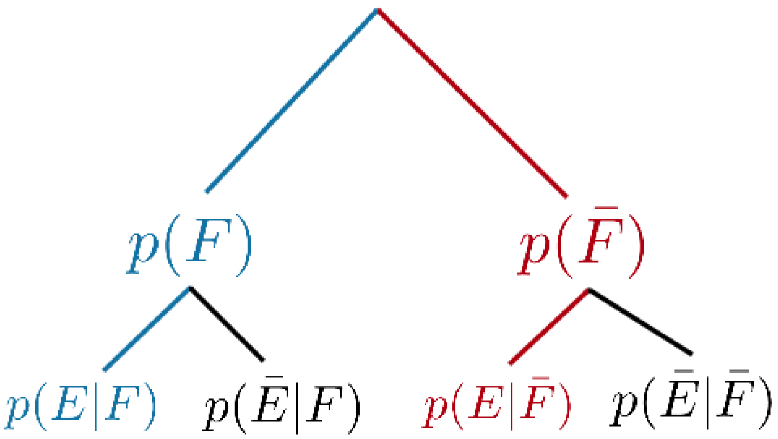
Putting everything together, we get our theorem:

$$p(F|E) = \frac{p(E|F)p(F)}{p(E|F)p(F) + p\big(E|\overline{F}\big) + p\big(E|\overline{F}\big)p\big(\overline{F}\big)}$$

□

Personal note

I personally draw a tree when I have to find this theorem back:



By definition of the conditional probabilities, we know that:

$$p(F|E) = \frac{p(E \cap F)}{p(E)}$$

$E \cap F$ can only be reached in one way: the blue way. Indeed, it is the only one that has both the positive version of E and F. So, $p(E \cap F) = p(F)p(E|F)$:

$$p(F|E) = \frac{p(F)p(E|F)}{p(E)}$$

To reach $p(E)$, there are two ways: the blue way and the red way. We can simply add the two paths since they are disjoint, so:

$$p(F|E) = \frac{p(F)p(E|F)}{p(E|F)p(F) + p\left(E|\overline{F}\right)p\left(\overline{F}\right)}$$

Finally, note that this theorem is so important it led to a philosophical movement, Bayesianism.

Example 1

Let's suppose that 2% of the population has Covid-19. There is a test for this disease that gives a positive result 95% of the time when given to someone with the disease. When given to someone without the disease, the tests gives a negative result 98% of the time. Note that the professor took this data some time ago, so they may not be very accurate nowadays.

Let D be the event that the person has the disease, and P the event that the person tests positive. We know that:

$$p(D) = 0.02, \quad p\left(\overline{D}\right) = 0.98, \quad p(P|D) = 0.95, \quad p\left(P|\overline{D}\right) = 0.02$$

We want to know the probability that a person who tests positive has Covid. Let's compute $p(D|P)$ first:

$$p(D|P) = \frac{p(P|D)p(D)}{p(P|D)p(D) + p\left(P|\overline{D}\right)p\left(\overline{D}\right)} = \frac{0.95 \cdot 0.02}{0.95 \cdot 0.02 + 0.02 \cdot 0.98} \approx 0.49$$

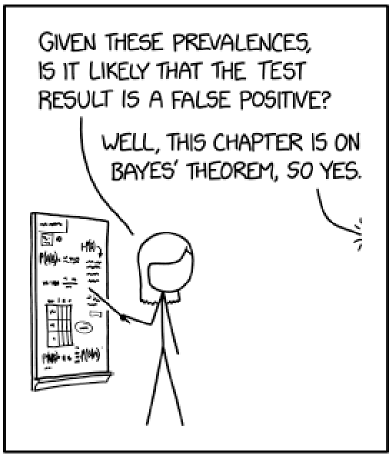
So, $p\left(\overline{D}|P\right) = 0.51$, which is very low. This basically means that half the person who get tested positive actually do not have the Covid.

Now, we want to know the probability that a person who tests negative does not have Covid:

$$p\left(D|\overline{P}\right) = \frac{p\left(\overline{P}|D\right)p(D)}{p\left(\overline{P}|D\right)p(D) + p\left(\overline{P}|\overline{D}\right)p\left(\overline{D}\right)} = \frac{0.05 \cdot 0.02}{0.05 \cdot 0.02 + 0.98 \cdot 0.98} \approx 0.001$$

So, if we test negative then the probability to have the virus is, in fact, really low. In other words, if we test positive, we don't really know whether we have the virus, but if we test negative, we can be quite sur that we do not have it.

Personal note Here is a relevant XKCD:


GIVEN THESE PREVALENCES, IS IT LIKELY THAT THE TEST RESULT IS A FALSE POSITIVE?
WELL, THIS CHAPTER IS ON BAYES' THEOREM, SO YES.

SOMETIMES, IF YOU UNDERSTAND BAYES' THEOREM WELL ENOUGH, YOU DON'T NEED IT.

https://xkcd.com/2545/

Example 2 Let's say we have two boxes. The first box contains two green balls and seven red balls. The second contains four green balls and three red balls.

Bob (who wants to add a letter in the middle of his name (that's a joke from Brandon Sanderson)) selects one of the boxes at random, then he selects a ball from that box at random. Knowing that he took a red ball, we want to know what is the probability that he chose the first box.

Our evidence is E, it is the event that he took a red ball. We know $p(E) = \frac{10}{16}$ since there is a total of 16 balls, amongst which 10 are red. The unknown probability is F, which is the event of choosing box 1. We have that $p(F) = \frac{1}{2}$. The likelihoods are given by:

$$p(E|F) = \frac{7}{9}, \quad p\left(E|\overline{F}\right) = \frac{3}{7}$$

We want to get the posterior probability:

$$p(F|E) = \frac{p(E|F)p(F)}{p(E|F)p(F) + p\left(E|\overline{F}\right)p\left(\overline{F}\right)} = \frac{\frac{7}{9}\cdot\frac{1}{2}}{\frac{7}{9}\frac{1}{2} + \frac{3}{7}\frac{1}{2}} = 0.645$$

It makes sense that it is more likely to have chosen the first box, since there are more red balls in it.

Generalised Bayes' theorem Suppose that E is an event from a sample space S such that $p(E) \neq 0$ and that $F_1, F_2, \ldots, F_n$ are mutually exclusive events such that:

$$\bigcup_{i=1}^{n} F_i = S$$

Then:

$$p(F_j|E) = \frac{p(E|F_j)p(F_j)}{p(E)} = \frac{p(E|F_j)p(F_j)}{\sum\limits_{i=1}^{n} p(E|F_i)p(F_i)}$$

Personal note We can do the exact same tree as above, with more branches. It would have more red branches, but sill one blue branch.

Monty Hall puzzle In a TV-game, there are three doors. Behind one of them, there is a car, and goats behind the others. The player first selects a door. Then, the host opens one of the

two other doors (they opens one that has a goat behind). Finally, the player can switch the door they chose. We wonder if they should switch.
Let pr be the probability that the initial choice has the car. The probability of winning without switching is given by:

$$p(win_1) = p(pr) = \frac{1}{3}$$

Now, if we switch, we have $p(win_2|pr) = 0$ and $p(win_2|\overline{pr}) = 1$ (if we had not chosen the right door first, we would be switching to the only closed door, which would have the price since the host opened the other door with a goat). So:

$$p(win_2) = p(win_2|pr)p(pr) + p(win_2|\overline{pr})p(\overline{pr}) = 0 \cdot \frac{1}{3} + 1 \cdot \frac{2}{3} = \frac{2}{3}$$

So, we are twice as lucky to win the game if we decide to switch.

13.7 Random variables

Motivation Often, we are not interested in the outcomes of an experiment, but some function of those outcomes. For example, at a casino, we may want to know how much money we would win by rolling two dice and computing their sum.

Definition: random variables A **random variable** X is a fonction $X : S \mapsto \mathbb{R}$ from the sample space S of an experiment to the set of real numbers $\mathbb{R}$.
Note that random variables have nothing to do with variables, and they are not random; they are functions.
The name comes from the 1940s. W. Feller and J. L. Doob flipped a coin to know whether they would use "random variable" or "chance variable"; Feller won and the term "random variable" has been used ever since.

Example 1 Let's suppose that a coin is flipped three times. Let $X(t)$ be the random variable that equals the number of heads that appears when t is the outcome. Our sample space is $S = \{H, T\}^3$.
We have:

$$X(TTT) = 0$$
$$X(HTT) = X(THT) = X(TTH) = 1$$
$$X(HHT) = X(HTH) = X(THH) = 2$$
$$X(HHH) = 3$$

Example 2 We want to describe the number of points obtained for answering a question in a MCQ from the exam.
Our sample space is $S = \{1, 2, 3, 4\}$, depending on the answer we check, and $\sum_{s \in S} p(s) = 1$. For example, if we are guessing:

$$p(s) = \frac{1}{4}$$

The random variable is given by:

$$X(s) = \begin{cases} 1 \text{ if the answer is correct} \\ -\dfrac{1}{3} \text{ if the answer is wrong} \end{cases}$$

If two answers are chosen, then:

$$S = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}, \quad p(s) = \frac{1}{6}$$

In this case, the random variable is:

$$X(s) = \begin{cases} \frac{1}{2} \text{ if one of the answers is correct} \\ -\frac{1}{2} \text{ if they are all wrong} \end{cases}$$

Wednesday 15th December 2021 — Lecture 26 : Reality vs. expectations

Definition: distribution of a random variable

The **distribution** of a random variable X on a sample space S is the set of pairs $(r, p(X = r))$ for all $r \in X(S)$, where $p(X = r)$ is the probability that X takes the value r:

$$p(X = r) = \sum_{s \in S, X(s) = r} p(s)$$

[put sum teacher to show that this adds up to 1 and thus that it is a probability distribution].

Probability mass function

If the range of the function X is countable, then $p(X = r)$ can be interpreted as a function: $p : X(S) \mapsto \mathbb{R}$. The, this function is called **probability mass function** and it is a probability distribution over the sample space $X(S)$. Indeed, we can see it sums to one:

$$\sum_{r \in X(S)} p(X = r) = \sum_{r \in X(S)} \sum_{s \in S, X(s) = r} p(s) = \sum_{s \in S} p(s) = 1$$

If the range of X is not countable, then it breaks down (the probability to get any number when generating a random real number between 0 and 10 is 0, but then the sum of all probabilities is 0, which is a problem). This is continuous probability and we use integrals for that.

In our case, we will only consider random variables with countable range, so we can definitely think of p as a function for now.

Example 1

Let's suppose that a coin is flipped three times. Let $X(t)$ be the random variable that equals the number of heads that appear when t is the outcome. Our sample space is $S = \{H, T\}^3$. We can compute the probability distribution of our random variable:

$$p(X = 0) = \sum_{s \in S, X(s) = 0} p(s) = \frac{1}{8}$$

$$p(X = 1) = \sum_{s \in S, X(s) = 1} p(s) = \frac{3}{8}$$

Continuing that way, we get that our probability distribution is:

$$\left\{\left(0, \frac{1}{8}\right), \left(1, \frac{3}{8}\right), \left(2, \frac{3}{8}\right), \left(3, \frac{1}{8}\right)\right\}$$

Example 2

Let's get back to measuring the number of points we get in exams. Let's consider the case in which we check two answers. Then:

$$S = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$$

$$X_2(s) = \begin{cases} \frac{1}{2} \text{ if the correct answer is ticked} \\ -\frac{1}{2} \text{ if the correct answer is not ticked} \end{cases}$$

Let's assume that the answer is randomly guessed:

$$p\left(X_2 = \frac{1}{2}\right) = \frac{1}{2}, \quad p\left(X_2 = \frac{-1}{2}\right) = \frac{1}{2}$$

Now, let's assume that one answer can be excluded. And that that we choose two answers out of the three left:

$$p\left(X_2 = \frac{1}{2}\right) = \frac{2}{3}, \quad p\left(X_2 = \frac{-1}{2}\right) = \frac{1}{3}$$

Finally, let's assume that two answers can be safely excluded:

$$p\left(X_2 = \frac{1}{2}\right) = 1, \quad p\left(X_2 = -\frac{1}{2}\right) = 0$$

Example 3: Bernoulli Trials

The number of successes in n Bernoulli trials is:

$$C(n,k)p^k q^{n-k} = b(k : n, p)$$

We can interpret $b(k : n, p)$ as a probability distribution:

$$p(X = k) = b(k : n, p)$$

13.8 Expected value

Definition: Expected value

The **expected value** (or **expectation** or **mean**) of the random variable X on the sample space S is equal to:

$$E(X) = \sum_{s \in S} p(s)X(s)$$

Example 1

Let X be the number that comes up when a fair dice is rolled. We want to compute the expected value of X:

$$E(X) = \sum_{s \in S} p(s)X(s) = \frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = \frac{7}{2}$$

Example 2

We wonder what is the expected value for getting points when guessing the answer. If we tick one answer, then the expected value is:

$$E(X_1) = \frac{1}{4} \cdot 1 + \frac{3}{4}\left(-\frac{1}{3}\right) = 0$$

Similarly, if we tick two answers:

$$E(X_2) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2}\left(-\frac{1}{2}\right) = 0$$

So, if we don't know anything, we basically don't get any points. This makes more sense than getting some points when knowing nothing, which then needs to be corrected.

Theorem

If X is a random variable and $p(X = r)$ is the probability distribution, where $p(X = r) = \sum_{s \in S, X(s) = r} p(s)$, then:

$$E(X) = \sum_{r \in X(S)} p(X = r)r$$

Proof By definition, our expected value is given by:

$$E(X) = \sum_{s \in S} p(s)X(s)$$

We can group terms:

$$\begin{aligned} E(X) &= \sum_{r \in X(S)} \sum_{s \in S, X(s) = r} p(s)X(s) \\ &= \sum_{r \in X(s)} \sum_{s \in S, X(s) = r} p(s)r \\ &= \sum_{r \in X(s)} r \sum_{s \in S, X(s) = r} p(s) \end{aligned}$$

We can recognise the definition of probability distribution of a random variable, so:

$$E(X) = \sum_{r \in X(S)} p(X = r)r$$

□

Example Let's say we are rolling a pair of fair dice. We want to know the expected value for the sum.
Let X be the random variable that is equal to the sum the dice. The range of X is $\{2, 3, \ldots, 11, 12\}$. Moreover, we can compute probabilities:

$$p(X = 2) = p(X = 12) = \frac{1}{36}$$

$$p(X = 3) = p(X = 11) = \frac{2}{36}$$

We can continue, and this allows us to get a sum of eleven terms:

$$E(X) = 2\frac{1}{36} + 3\frac{2}{36} + \ldots + 11\frac{2}{36} + 12\frac{1}{36} = 7$$

If we had considered the whole domain, we would have had thirty-six terms in this sum!

Theorem: Expected value of Bernoulli trials The expected number of successes when n mutually independent Bernoulli trials are performed is np, where p is the probability of success of each trial.

Proof We could prove this theorem my using our theorem above, and the following equality:

$$k\binom{n}{k} = n\binom{n-1}{k-1}$$

However, this would take around 5 lines, whereas we will be able to prove this theorem in one line next week (it is on the next page in this document, but for me it is next week (in fact, I am cleaning those notes on Saturday the 18th of December, so it's not exactly in a week, but you get what I mean)).

Tuesday 21st December 2021 — **Lecture 27 : Staying on the road on average**

Example

On an exam we have 24 questions with 4 choices. Let's say we are doing random guessing. We want to know the expected value.
This is a Bernoulli trial with $n = 24$ and $p = \frac{1}{4}$, so:

$$E(X) = \frac{24}{4} = 6$$

Similarly, if someone knows the answer to 12 questions, then, by guessing randomly on the 12 other questions, he or she will get $E(X) = 3$ points. So, if the teacher wants us to answer 12 questions, then the average must be at 15 points.

Theorem: linearity of expectations

Let $X_1, \ldots, X_n$ are random variables on S and $a, b \in \mathbb{R}$. Then:

$$E(X_1 + \ldots + X_n) = E(X_1) + \ldots + E(X_n)$$

$$E(aX + b) = aE(X) + b$$

Proof

The proof is based on the fact that we are using sums, which are linear.
Let's proof the case that $E(X_1 + X_2) = E(X_1) + E(X_2)$. Notice that X_1 and X_2 are functions, so $(X_1 + X_2)(s) = X_1(s) + X_2(s)$. Thus, by our theorem on the computation of expected values:

$$E(X_1 + X_2) = \sum_{s \in S} (X_1 + X_2)(s) p(s) = \sum_{s \in S} (X_1(s) + X_2(s)) p(s)$$

Which we can simplify using the linearity of summations:

$$\sum_{s \in S} X_1(s) p(s) + \sum_{s \in S} X_2(s) p(s) = E(X_1) + E(X_2)$$

Example 1

We are rolling two dice. Let X_1 be the number that the first dice rolled, and X_2 be the same for the second dice. We know that:

$$E(X_1) = E(X_2) = \frac{7}{2}$$

Thus, by our theorem, the expected value of their sum is given by:

$$E(X_1 + X_2) = E(X_1) + E(X_2) = \frac{7}{2} + \frac{7}{2}$$

In other words, we did not have to do all the hard work we did to compute the expected value of the sum of two dice; this theorem is very powerful.

Example 2: Bernoulli Trials

Let's say we are doing Bernoulli trials with probability of success p. Let X_i be the random variable that gives 1 if the i^{th} trial is a success, and 0 if it is not. We notice that the expected value of a specific variable is:

$$E(X_i) = p \cdot 1 + 0(1 - p) = p$$

Thus, the expected value of n trials is given by:

$$E(X_1 + \ldots + X_n) = E(X_1) + \ldots + E(X_n) = p + \ldots + p = np$$

Example 3

Let X_i be the random variable that defines the number of points we get when checking i answers:

$$X_1(S) = \begin{cases} 1, & \text{correct} \\ \frac{-1}{3}, & \text{incorrect} \end{cases} \qquad X_2(S) = \begin{cases} \frac{1}{2}, & \text{correct} \\ \frac{-1}{2}, & \text{incorrect} \end{cases}$$

Let's say we no longer want negative points, so let us define $Y_i(s) = 6X_i(s) + 3$:

$$Y_1(S) = \begin{cases} 9, & \text{correct} \\ 1, & \text{incorrect} \end{cases} \qquad Y_2(S) = \begin{cases} 6, & \text{correct} \\ 0, & \text{incorrect} \end{cases}$$

We can compute the expected value of our new variables:

$$E(Y_i) = 6E(X_i) + 3 = 6 \cdot 0 + 3 = 3$$

Which is a bias the Professor would have to correct when setting the average.

Definition The ordered pair (i, j) is an **inversion** of a permutation of the first n positive integers if $i < j$ but j precedes i in the permutation.

> *Example* There are six inversions in the permutation of $3, 5, 1, 4, 2$:
>
> $$(1, 3), (1, 5), (2, 3), (2, 4), (2, 5), (4, 5)$$

Expected number of inversions We want to find the expected number of inversions in a random permutation of the first n integers.

Let S be our sample space, i.e. all permutations of n integers. Moreover, let $I_{i,j}(s)$ be a random variable defined as follows:

$$I_{i,j}(s) = \begin{cases} 1, & \text{if } (i, j) \text{ has been inverted} \\ 0, & \text{otherwise} \end{cases}$$

Let's also define the following random variable:

$$X(s) = \sum_{1 \leq i < j \leq n} I_{i,j}(s)$$

It counts the number of inversions. Let's compute the expected value of $I_{i,j}$:

$$E(I_{i,j}) = \sum_{r \in I_{i,j}(S)} rp(I_{i,j} = r) = 1 \cdot p(I_{i,j} = 1) + 0 \cdot p(I_{i,j} = 0) = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = \frac{1}{2}$$

Indeed, if we fix a pair, then for every permutation there are two versions: one in which they are exchanged and one in which they are in order. Thus, $p(I_{i,j} = 1) = p(I_{i,j} = 0) = \frac{1}{2}$.

We still need to compute the expected value of X:

$$E(X) = E\left(\sum_{1 \leq i < j \leq n} I_{i,j}\right) = \sum_{1 \leq i < j \leq n} E(I_{i,j}) = \sum_{1 \leq i < j \leq n} \frac{1}{2} = \binom{n}{2}\frac{1}{2} = \frac{n(n-1)}{4}$$

Definition: Independent random variables The random variables X and Y on a sample space S are independent if:

$$p(X = r_1 \land Y = r_2) = p(X = r_1) \cdot p(Y = r_2)$$

Theorem If X and Y are independent variables on a sample space S, then:

$$E(X \cdot Y) = E(X)E(Y)$$

Proof

Let's do a direct proof:

$$E(X \cdot Y) = \sum_{s \in S} (X \cdot Y)(s) p(s) = \sum_{s \in S} X(s) Y(s) p(s)$$

As usual, let's group terms:

$$\sum_{r_1 \in X(S)} \sum_{r_2 \in X(S)} \sum_{\substack{s \in S \\ X(s) = r_1 \\ Y(s) = r_2}} r_1 r_2 p(s) = \sum_{r_1 \in X(S)} \sum_{r_2 \in X(S)} r_1 r_2 \sum_{\substack{s \in S \\ X(s) = r_1 \\ Y(s) = r_2}} p(s)$$

The third sum is, by definition, $p(X = r_1 \land Y = r_2)$:

$$\sum_{r_1 \in X(S)} \sum_{r_2 \in X(S)} r_1 r_2 p(X = r_1 \land Y = r_2)$$

We know that our two random variables are independent, so:

$$\sum_{r_1 \in X(S)} \sum_{r_2 \in X(S)} r_1 r_2 p(X = r_1) p(Y = r_2)$$

Again, we can group terms:

$$\sum_{r_1 \in X(S)} r_1 p(X_1 = r_1) \underbrace{\sum_{r_2 \in Y(s)} r_2 p(Y = r_2)}_{E(Y)}$$

Since $E(Y)$ does not depend on r_1, we can factor it out of the sum:

$$E(Y) \sum_{r_1 \in X(S)} r_1 p(X = r_1) = E(Y) E(X)$$

□

13.9 Variance

Definition: variance Let X be a random variable on the sample space S. The **variance** of X, denoted by $V(X)$, is:

$$V(X) = \sum_{s \in S} (X(s) - E(X))^2 p(s)$$

Motivation There are many ways of having a random variable which expected value is a number. For instance, it could always give the same number, or it could give numbers in both extremes. As the Professor says, if you are on the road on average, it does not make you a good driver.

Variance is a way of measuring the "average distance to the mean". The square is here for many reasons. First, it makes $X(S) - E(X)$ positive (working with absolute values is always very complicated; squares allow us to differentiate this, for example). Second, this is usual to take the square of the distance (see least squares in algebra, for example). Third, this will give us a nice theorem hereinafter.

Definition: standard deviation The **standard deviation** of X, denoted by $\sigma(X)$, is defined as:

$$\sigma(X) = \sqrt{V(X)}$$

Note We take a square root to remove the square from the variance. This is for example very important for physical units.

Example Let X and Y be random variables on $S = \{1, 2, 3, 4, 5, 6\}$. Let's define them as follows:

$$X(s) = 0, \quad \forall s \in S \qquad\qquad Y(s) = \begin{cases} -1, & s \in \{1, 2, 3\} \\ 1, & s \in \{4, 5, 6\} \end{cases}$$

We can compute their expected values, and realise that, in fact, we cannot distinguish them thanks to that:

$$E(X) = E(Y) = 0$$

Let's now compute the variance of X:

$$V(X) = \sum_{s \in S} \underbrace{(X(s) - E(X))^2}_{=0} p(s) = 0$$

Let's also compute the variance of Y:

$$V(Y) = \sum_{s \in S} \underbrace{(Y(s) - E(Y))^2}_{=1} p(s) = \sum_{s \in S} \frac{1}{6} = 1$$

Theorem If X is a random variable on a sample space S, then:

$$V(X) = E\left(X^2\right) - E(X)^2$$

Proof By definition of the variance:

$$\begin{aligned} V(X) &= \sum_{s \in S} (X(s) - E(X))^2 p(s) \\ &= \underbrace{\sum_{s \in S} X(s)^2 p(s)}_{=E(X^2)} - 2 \sum_{s \in S} X(s) E(X) p(s) + \sum_{s \in S} E(X)^2 p(s) \\ &= E\left(X^2\right) - 2E(X) \underbrace{\sum_{s \in S} X(s) p(s)}_{=E(X)} + E(X)^2 \underbrace{\sum_{s \in S} p(s)}_{=1} \\ &= E\left(X^2\right) - 2E(X)E(X) + E(X)^2 \\ &= E\left(X^2\right) - E(X)^2 \end{aligned}$$

□

Corollary If X is a random variable on a sample space S, and $E(X) = \mu$, then:

$$V(X) = E\left((X - \mu)^2\right)$$

Proof Let's look at $E\left((X - \mu)^2\right)$:

$$E\left((X - \mu)^2\right) = E\left(X^2 - 2\mu X + \mu^2\right) = E\left(X^2\right) - 2\mu E(X) + \mu^2$$

And thus, by the definition of μ:

$$E\left(X^2\right) - 2\mu\mu + \mu^2 = E\left(X^2\right) - \mu^2 = E\left(X^2\right) - E(X)^2 = V(X)$$

Which is the variance by our theorem above.

□

Theorem: law of the unconscious statistician Let X be a random variable, and g be a function. Then:

$$E(g(X)) = \sum_{x \in X(S)} g(x)p(X = x)$$

Name This theorem is named the "law of the unconscious statistician" since we apply it very often without realising that we are doing so.

Proof Let's do a direct proof:

$$E(g(X)) = \sum_{s \in S} g(X(s))p(s) = \sum_{x \in X(S)} \sum_{\substack{s \in S \\ X(s) = x}} g(X(s))p(s)$$

Since we grouped terms, $X(S) = x$, which does not depend on s:

$$\sum_{x \in X(S)} g(x) \sum_{\substack{s \in S \\ X(s) = x}} p(s) = \sum_{x \in X(S)} g(x)p(X = x)$$

□

Example Let's compute the variance of the number that comes up when a fair die is rolled:

$$V(X) = E\left(X^2\right) - E(X)^2$$

We already know that $E(X) = \frac{7}{2}$, but we must also compute $E\left(X^2\right)$. We can do so by using the law of the unconscious statistician:

$$E\left(X^2\right) = \frac{1}{6}\left(1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2\right) = \frac{91}{6}$$

So, we get that:

$$V(X) = \frac{91}{6} - \left(\frac{7}{2}\right)^2 = \frac{35}{12}$$

Bienaymé's formula Let X and Y be two independent random variables on a sample space S, then:

$$V(X + Y) = V(X) + V(Y)$$

This can be generalised to n independent random variables:

$$V(X_1 + \ldots + X_n) = V(X_1) + \ldots + V(X_n)$$

Proof Let's do a direct proof:

$$\begin{aligned} V(X + Y) &= E\left((X + Y)^2\right) - E(X + Y)^2 \\ &= E\left(X^2 + 2XY + Y^2\right) - (E(X) + E(Y))^2 \\ &= E\left(X^2\right) + 2 \overbrace{E(XY)}^{= E(X)E(Y)} + E\left(Y^2\right) \\ &\quad - E(X)^2 - 2E(X)E(Y) - E(Y)^2 \\ &= E\left(X^2\right) - E(X)^2 + E\left(Y^2\right) - E(Y)^2 \\ &= V(X) + V(Y) \end{aligned}$$

So, we are indeed using the fact that those random variables are independent.

□

Variance of Bernoulli trials

We want to know the variance of random variable X, where $X(t) = 1$ if the Bernoulli trial is a success, and $X(t) = 0$. We know that $E(X) = p$, let's compute $E\left(X^2\right)$ using the law of the unconscious statistician:

$$E\left(X^2\right) = p \cdot 1^2 + (1-p) \cdot 0^2 = p$$

So, our variance is given by:

$$V(X) = E\left(X^2\right) - E(X)^2 = p - p^2 = p(1-p) = pq$$

Thus, since trials are independent, the variance of n of them is given by:

$$V(X_1 + \ldots + X_n) = V(X_1) + \ldots + V(X_n) = pq + \ldots + pq = npq$$

Wednesday 22nd December 2021 — **Lecture 28 : My exam went rather well!**

13.10 Inequalities for deviation

Notation

We note $p(X \geq a)$ to say "the probability that X is greater than or equal to a. This is equivalent to:

$$p(A), \quad A = \{s \in S | X(s) \geq a\}$$

Markov's Inequality

Let X be a non-zero and non-negative random variable, i.e.:

$$\exists s \; X(s) > 0 \land \forall s \; X(s) \geq 0$$

Then, we have the following inequality for all $M > 0$:

$$p(X \geq ME(X)) \leq \frac{1}{M}$$

Proof We can start from our lower bound:

$$\frac{1}{M} = \frac{E(X)}{ME(X)} = \frac{\sum\limits_{x \in X(S)} xp(X = x)}{ME(X)} \geq \frac{\sum\limits_{\substack{x \in X(S) \\ x \geq ME(X)}} xp(X = x)}{ME(X)}$$

Which is greater than or equal to:

$$\frac{\sum\limits_{\substack{x \in X(S) \\ x \geq ME(X)}} ME(X)p(X = x)}{ME(X)} = \sum_{\substack{x \in X(S) \\ x \geq ME(X)}} p(X = x) = p(X \geq ME(X))$$

Example

Let's say we are rolling a die 6 times. We consider rolling a 6 as a success.
We know that the expected number of success over those 6 trials is $E(X) = 1$. We can estimate the probability of having at least 3 successes using Markov inequality:

$$p(X \geq 3) = p(X \geq 3E(X)) \leq \frac{1}{3}$$

We will do the computations using Bernoulli trials, and see that this is actually not very efficient.

Chebyshev's Inequality

Let X be a random variable on a sample space S with probability function p. If r is a positive real number, then:

$$p(|X(s) - E(X)| \geq r) \leq \frac{V(X)}{r^2}$$

Proof Let A be the event defined as follows:

$$A = \{s \in S \text{ such that } |X(S) - E(X)| \geq r\}$$

So, we can see that:

$$p(A) = p(|X(s) - E(X)| \geq r)$$

$$V(X) = \sum_{s \in S} (X(s) - E(X))^2 p(s)$$

by definition of the variance.
Let's work on the second equality:

$$\begin{aligned} V(X) = & \sum_{s \in S} (X(s) - E(X))^2 p(s) \\ = & \sum_{s \in A} (X(s) - E(X))^2 p(s) + \underbrace{\sum_{s \in S \setminus A} (X(s) - E(X))^2 p(s)}_{\geq 0} \\ \geq & \sum_{s \in A} (X(s) - E(X))^2 p(s) \\ \geq & \sum_{s \in A} r^2 p(s) = r^2 \sum_{s \in A} p(s) = r^2 p(|X(s) - E(X)| \geq r) \end{aligned}$$

So, we deduce that:

$$p(|X(s) - E(X)| \geq r) \leq \frac{V(X)}{r^2}$$

□

Example Again, we are rolling 6 times a die and consider rolling a 6 being a success.
We can see that those are Bernoulli trials, with $n = 6$ and $p = \frac{1}{6}$. Moreover, we know that:

$$E(X) = np = \frac{6}{6} = 1, \quad V(X) = npq = 6 \cdot \frac{1}{6} \cdot \frac{5}{6} = \frac{5}{6}$$

So, using Chebyshev's inequality, we can estimate the probability of having at least 3 successes:

$$p(|X(s) - E(X)| \geq 2) \leq \frac{V(X)}{2^2} = \frac{5}{6 \cdot 2^2} = \frac{5}{24}$$

We notice that this is a smaller probability than the $\frac{1}{3}$ we got with Markov inequality.
We could also compute the exact probability of having 3 or more successes using the binomial distribution:

$$b\left(3 : 6, \frac{1}{6}\right) + b\left(4 : 6, \frac{1}{6}\right) + b\left(5 : 6, \frac{1}{6}\right) + b\left(6 : 6, \frac{1}{6}\right)$$

Which is equal to:

$$\frac{6 \cdot 5 \cdot 4 \cdot 5^3}{1 \cdot 2 \cdot 3 \cdot 6^6} + \frac{5 \cdot 5 \cdot 5^2}{1 \cdot 2 \cdot 6^6} + \frac{6 \cdot 5}{1 \cdot 6^6} + \frac{1}{6^6} \approx 0.06$$

So, we can see that none of these inequalities is very efficient. However, they are still estimations that can be useful if we don't know anything.

13.11 Geometric distribution

Example Let's take a biased coin which probability of having tail (T) is p.
We wonder what is the number of tosses needed until a tail shows up. Our sample space is given by:

$$S = \{T, HT, HHT, HHHT, \ldots\} \implies |S| = \aleph_0$$

So, we can compute the probabilities:

$$p(T) = p, \quad p(HT) = (1-p)p, \quad p(HHT) = (1-p)^2 p, \quad \ldots$$

More generally:

$$p\left(H^k T\right) = (1-p)^k p$$

Let's define $X(s)$ as the number of toss needed to have a tail. So:

$$p(X = k) = (1-p)^k p$$

Definition Let X be a random variable counting the number of failures of an experiment before the first success (which occurs at probability p). Then:

$$p(X = k) = (1-p)^k p$$

Equivalently, if Y counts the number of trials until the success (included), we have $Y = X + 1$, so:

$$p(Y = k) = (1-p)^{k-1} p$$

This is called the **geometric distribution**.

> *Example* Using this distribution, we could consider the length of someone's life if the probability of dying each years is constant.

Expected value We can compute the expected value:

$$E(X) = \sum_{r \in X(S)} r p(X = r) = \sum_{j=1}^{\infty} j p(X = j) = \sum_{j=1}^{\infty} j (1-p)^{j-1} p = p \sum_{j=1}^{\infty} j (1-p)^{j-1}$$

There are two ways of computing a sum looking that way. The first one is to remember that, when using generating functions, we saw that:

$$\frac{1}{(1-x)^2} = \sum_{j=0}^{\infty} C(2 + j - 1, j) x^j = \sum_{j=0}^{\infty} \binom{j+1}{j} x^j = \sum_{j=0}^{\infty} (j+1) x^j = \sum_{j=1}^{\infty} j x^{j-1}$$

The second one is to differentiate:

$$\sum_{j=0}^{\infty} x^j = \frac{1}{1-x} \overset{\frac{d}{dx}}{\implies} \sum_{j=1}^{\infty} j x^{j-1} = \frac{1}{(1-x)^2}$$

So, using what we just found:

$$E(X) = p \frac{1}{(1-(1-p))^2} = \frac{p}{p^2} = \frac{1}{p}$$

Variance By one of our theorems, we know that:

$$V(X) = E\left(X^2\right) - E(X)^2 = E(X(X-1) + X) - E(X)^2 = E(X(X-1)) + E(X) - E(X)^2$$

We know everything but $E(X(X-1))$. We can compute it using the law of the unconscious statistician:

$$E(X(X-1)) = \sum_{j=1}^{\infty} j(j-1)p(X=j) = \sum_{j=1}^{\infty} j(j-1)(1-p)^{j-1}p = \sum_{j=0}^{\infty}(j+1)j(1-p)^{j}p$$

So:

$$E(X(X-1)) = 2p(1-p)\sum_{j=0}^{\infty}\frac{j(j+1)}{2}(1-p)^{j-1}$$

This sum can also be computed using derivatives, or using generating functions:

$$\frac{1}{(1-x)^3} = \sum_{j=0}^{\infty} C(3+j-1, j)x^j = \sum_{j=0}^{\infty}\binom{j+2}{j}x^j = \sum_{j=0}^{\infty}\frac{j(j+1)}{2}x^j$$

Which we can simplify to:

$$\frac{1}{(1-x)^3} = \sum_{j=1}^{\infty}\frac{(j+1)j}{2}x^{j-1} = \sum_{j=0}^{\infty}\frac{(j+1)j}{2}x^{j-1}$$

So, we can compute our variance:

$$V(X) = E(X(X-q)) + E(X) - E(X)^2 = \frac{2(1-p)}{p^2} + \frac{1}{p} - \frac{1}{p^2} = \frac{2-2p+p-1}{p^2}$$

Thus, simplifying our result:

$$V(X) = \frac{1-p}{p^2} = \frac{q}{p^2}$$

Using coins to generate random numbers

Let's assume that we have a fair coin, and that we would like to generate a random numbers in $\{0, \dots, n-1\}$.
The simple case is if $n-1 = 2^j$, with $j \geq 1$. Indeed, we only have to flip the coin j times, and consider the result as the number in its binary form. However, this is harder if n does not have this form.
If $n = 3$, let's use the following algorithm:

```
procedure random_number_3():
    while True:
        toss 2 coins
        if result is in [00, 01, 10]
            return result
```

This code could never end. The probability to get a number at the n^{th} roll is given by a geometric distribution, where $p = \frac{3}{4}$:

$$p(X=1) = \frac{3}{4}, \quad p(X=2) = \frac{1}{4}\cdot\frac{3}{4} = \frac{3}{16}, \quad \dots, \quad p(X=k) = \left(\frac{1}{4}\right)^k \cdot \frac{3}{4}$$

Moreover, we know that the expected number of iteration of the loop is:

$$E(X) = \frac{1}{p} = \frac{4}{3}$$

And the expected of number of tosses of coins is:

$$E(2X) = 2E(X) = \frac{8}{3}$$

However, this is not guaranteed, and it could run forever. So, we want to know how likely it is to have many tosses. Let's compute the variance:

$$V(X) = \frac{1-p}{p^2} = \frac{\frac{1}{4}}{\frac{9}{16}} = \frac{4}{9}$$

We want to know the probability that $X(s) \geq k$. We want to apply Chebychev inequality, but we need to modify our inequality first:

$$X(s) \geq k \iff X(s) - \frac{4}{3} \geq k - \frac{4}{3} \iff \left|X(s) - E(X)\right| \geq k - \frac{4}{3}$$

So, we can apply Chebychev inequality:

$$p(X(s) \geq k) = p\left(\left|X(s) - E(X)\right| \geq k - \frac{4}{3}\right) \leq \frac{V(X)}{\left(k - \frac{4}{3}\right)^2} = \frac{\frac{4}{9}}{\left(k - \frac{4}{3}\right)^2} = \frac{4}{\left(3k - 4\right)^2}$$

The specific parameters are not really important, but we can see that the probability falls at least as $\frac{1}{k^2}$, since it is at least $\Theta\left(\frac{1}{k^2}\right)$.

Advice from the Professor

Doing exercises is a good way to practice, since this allows us to revise (we will see which parts of theory we are lacking).