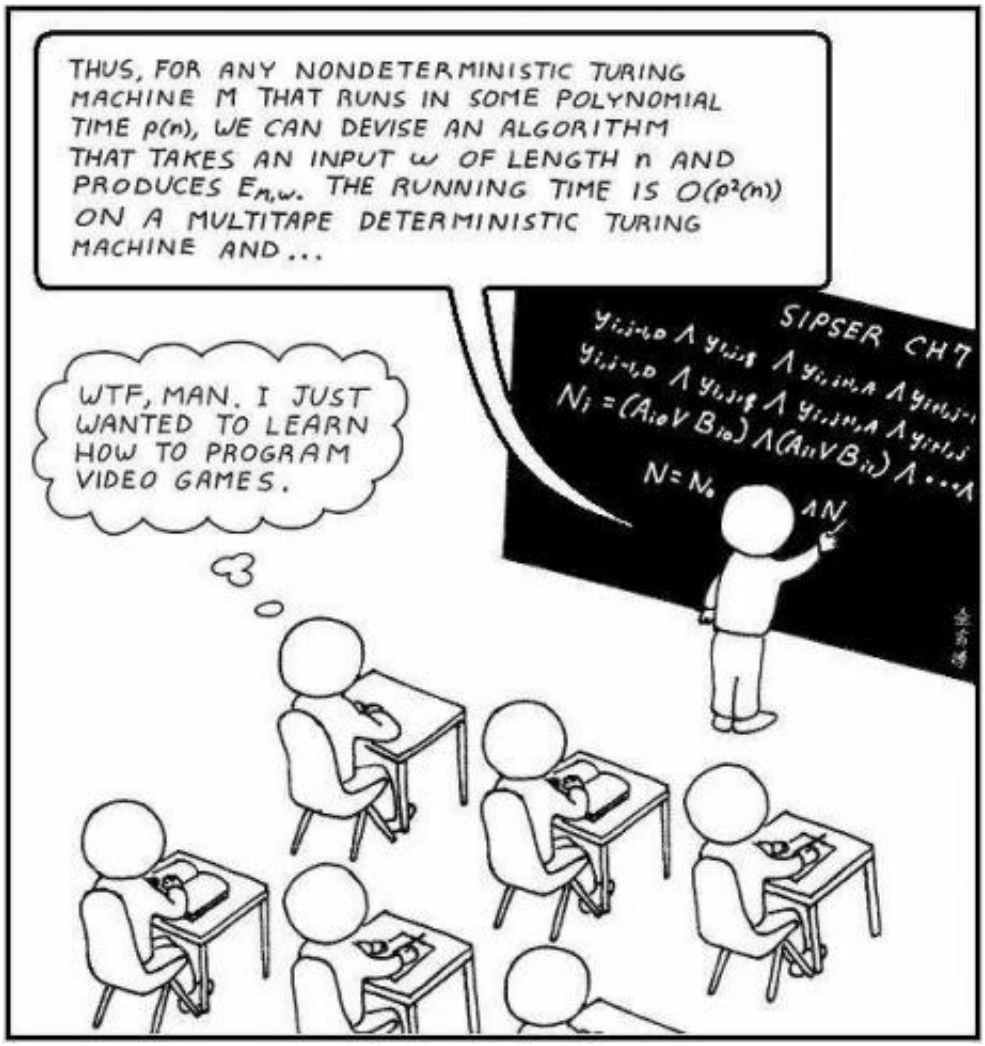
Mathématiques pour l'Informatique



Benjamin DUPONT
Licence Informatique & Vidéoludisme - Université Paris 8
2022-2023

Table des matières

Chapitre 1

Ensembles et logique

1.1. Ensembles et fonctions

Les ensembles sont des objets centraux des fondements de la plupart des théories mathématiques. Ils permettent de rassembler des objets de même nature afin de les étudier ainsi que les relations entre ces objets.

1.1.1 Definition. Plus formellement, un ensemble est une collection d'objets mathématiques, différents les uns des autres, appelés éléments de l'ensemble. Deux ensembles sont égaux, ou coïncident, si et seulement si ils ont les mêmes éléments.

La manière courant pour représenter un ensemble est la notation *en extension*, c'est-à-dire qu'un ensemble est décrit par des symboles d'accolades {}, et ses éléments sont listés et séparés par une virgule ,.

1.1.2 Exemple.
- Par exemple, $\{1,2,3,4,5\}$ est un ensemble dont les éléments sont les nombres $1,2,3,4$ et 5.
- $\{\text{vrai}, \text{faux}\}$ est un ensemble contenant deux éléments, appelés vrai et faux qui sont des *booléens*.

Dans un ensemble, l'ordre d'apparition n'a pas d'importance. Chaque élément n'apparaît qu'une seule fois. Dans une écriture en extension, si un élément apparaît plusieurs fois, alors cela ne change pas la définition de l'ensemble et il faut considérer que l'élément n'est contenu qu'une fois. Ainsi,

$$\{1,2,3,4,5\} = \{4,3,5,2,1\} = \{1,2,3,4,1,3,5\}.$$

On utilise le symbole $\in$ pour dire qu'un élément appartient à un ensemble, et $\notin$ pour dire qu'il n'appartient pas. Ainsi, $3 \in \{1,2,3,4,5\}$ et $6 \notin \{1,2,3,4,5\}$.

1.1.3 Remarque. Les ensembles étant aussi des objets mathématiques, on peut faire des ensembles d'ensembles. Ainsi, $\{1,\{2\},\{3,4\}\}$ est un ensemble contenant 3 éléments : le nombre 1, l'ensemble contenant un seul élément (le nombre 2) et l'ensemble contenant deux éléments (les nombres 3 et 4).

1.1.4 Definition. On dit qu'un ensemble A est un *sous-ensemble* d'un ensemble B si tous les éléments de A sont aussi dans B. On note alors $A \subset B$, et on dit que A est *inclus* dans B.

Si A n'est pas un sous-ensemble de B, on note $A \not\subset B$.

Il y a un ensemble qui ne contient également, on l'appelle *ensemble vide*, et il est noté $\emptyset$. Si un ensemble ne contient qu'un seul élément, on parle de *singleton*.

1.1.5 Exemple. On a $\{2,3\} \subset \{1,2,3,4,5\}$ mais $\{2,6\} \subset \{1,2,3,4,5\}$.

On a aussi $\emptyset \subset \{1,2,3,4,5\}$ tandis que $\{1,2,3,4,5\} \not\subset \emptyset$.

On peut aussi décrire un ensemble comme l'ensemble des éléments d'un autre ensemble satisfaisant une propriété supplémentaire. On utilise alors une barre verticale pour séparer l'ensemble de départ de la propriété qu'on ajoute. Par exemple,

$$\{x \in \{1,2,3,4,5\} \mid x \leq 3\}$$

est le sous-ensemble de $\{1,2,3,4,5\}$ contenant les nombres inférieurs ou égaux à 3. C'est donc l'ensemble $\{1,2,3\}$. La barre verticale dans cette notation a le sens de « satisfaisant ». Ainsi, on peut lire $\{x \in \{1,2,3,4,5\} \mid x \leq 3\}$ comme « l'ensemble des nombres x de $\{1,2,3,4,5\}$ satisfaisant $x \leq 3$ ».

Enfin, on peut décrire un ensemble en appliquant une fonction aux éléments d'un ensemble existant : on décrit d'abord la fonction appliquée à une variable, puis on met un point-virgule, puis on décrit l'ensemble de départ. On dit qu'on définit l'ensemble *en fonction*. Nous reviendrons plus en détails sur les notions de fonctions et de variable plus tard. Par exemple, l'ensemble

$$\{2n+1 \ ; \ n \in \{1,2,3\}\}$$

est l'ensemble obtenu en partant de $\{1,2,3\}$ et en appliquant la fonction $n \mapsto 2n+1$ sur tous les éléments. C'est donc l'ensemble $\{3,5,7\}$.

Dans cette notation, le point-virgule a le sens de « où ». Ainsi, on peut lire l'ensemble ci-dessus comme « l'ensemble des nombres de la forme $2n+1$ où n appartient à $\{1,2,3\}$. »

1.1.6 Definition. Soient A et B deux ensembles.

1. La *réunion* de A et B, notée $A \cup B$, est l'ensemble qui contient tous les éléments de A et tous les éléments de B.

2. L'*intersection* de A et B, notée $A \cap B$, est l'ensemble des éléments qui sont à la fois dans A et dans B.

3. La *différence* de A et B, notée $A - B$ est l'ensemble qui contient tous les éléments de A qui ne sont pas dans B. Lorsque B est un sous-ensemble de A, la différence $A - B$ est aussi appelée complémentaire de B dans A.

1.1.7 Exemple.

1. $\{1,2,3\} \cup \{3,4,5\} = \{1,2,3,4,5\}$.

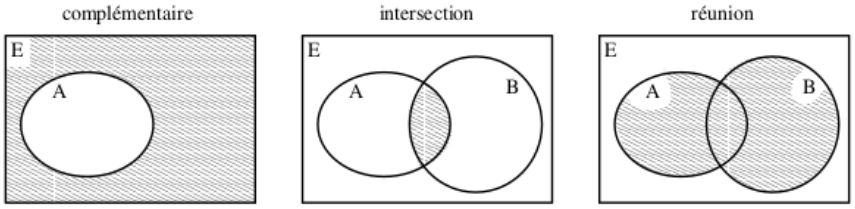
2. $\{1,2,3\} \cap \{3,4,5\} = \{3\}$.

3. $\{1,2,3,4\} - \{3,4,5\} = \{1,2\}$.

Quelques propriétés importantes:

- $A \cap B = B \cap A$,
- $A \cup B = B \cup A$,
- $A \cap (B \cap C) = (A \cap B) \cap C$,
- $A \cup (B \cup C) = (A \cup B) \cup C$,
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$,
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$,
- $E - (E - A) = A$,
- $E - (A \cup B) = (E - A) \cap (E - B)$,
- $E - (A \cap B) = (E - A) \cup (E - B)$.

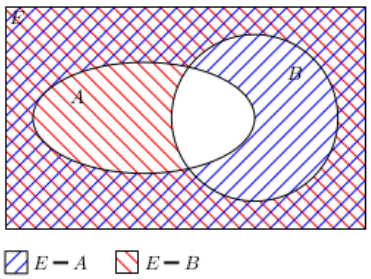
Pour mieux visualiser ces propriétés, il est commode de réaliser le gros ensemble E dans lequel on travaille comme un rectangle, et les sous-ensembles de E par des « patates » hachurées dessinées dans ce rectangle. Une telle représentation est appelé un *diagramme de Venn*, voir ci-dessous.



Les égalités ci-dessus peuvent se visualiser facilement sur un diagramme de Venn. Par exemple, voici ci-dessous un diagramme de Venn qui illustre la formule

$$(E - A) \cup (E - B) = E - (A \cap B).$$

Sur cette figure, seule l'intersection $A \cap B$ n'est pas hachurée.



Enfin, une dernière notation qui peut s'avérer utile : soit E un ensemble, tel qu'à chaque élément x de E on associe un ensemble E_x. Alors on définit les ensembles

$$\bigcup_{x \in E} E_x \qquad \left(\text{resp. } \bigcap_{x \in E} E_x \right)$$

comme la réunion (resp. l'intersection) de tous les ensembles E_x. Par exemple,

$$\bigcup_{x \in \{0,1,2\}} \{2x+1, 2x+2\} = \{1,2\} \cup \{3,4\} \cup \{5,6\} = \{1,2,3,4,5,6\},$$

$$\bigcap_{x \in \{0,1,2\}} \{x-1, x, x+1\} = \{-1,1,1\} \cap \{0,1,2\} \cap \{1,2,3\} = \{1\}.$$

1.1.8. Ensembles de nombres. Nous allons maintenant détailler certains ensembles très importants : les ensembles de nombres.

Nombres entiers naturels. Les nombres entiers naturels sont les nombres entiers positifs, à savoir $0, 1, 2$,etc. On note $\mathbb{N}$ l'ensemble de tous ces nombres, et $\mathbb{N}^*$ l'ensemble des nombres entiers naturels non nuls.

On rappelle qu'il y a deux opérations définies pour toutes les paires d'entiers naturels, addition et multiplication, notées $+$ et $\times$. Le nombre 0 est dit *neutre* vis-à-vis de l'addition, puisque pour tout nombre entier naturel n, on a $n + 0 = 0 + n = n$. De même, le nombre 1 est neutre vis-à-vis de la multipli ation, puisque pour tout nombre appelées entier naturel n on a $n \times 1 = 1 \times n = n$. Il y a sur les entiers naturels une relation d'ordre, notée $\leq$. Formellement il s'agit d'une fonction prenant deux entiers et rendant vrai ou faux selon que le premier nombre est inférieur ou égal au second, ou pas. Plutôt que de noter $\leq (x, y) =$ vrai, on note $x \leq y$, et plutôt que de noter $\leq (x, y) =$ faux, on note $x > y$. L'ordre $\leq$ satisfait $0 \leq 1$, et se comporte bien vis-à-vis de l'addition et la multiplication des entiers naturels : si a, b, c, d sont quatre entiers naturels, avec $a \leq b$ et $c \leq d$, on a $a + c \leq b + d$ et $a \times c \leq b \times d$. On note aussi $a < b$ si on a $a \leq b$ et $a \neq b$, et on note $a \geq b$ si on a $b \leq a$.

Notons aussi que l'ordre sur les entiers est *total*, ce qui signifie que deux entiers quelconques peuvent toujours être comparés. En effet, étant donnée $a, b \in \mathbb{N}$, on a toujours soit $a < b$, soit $a = b$, soit $a > b$. Lorsque a et b sont deux entiers naturels satisfaisant $b \leq a$, on peut définir leur différence, notée $a - b$, comme l'entier naturel c tel que $b + c = a$. La division est un peu spéciale : on ne peut diviser des entiers quelconques mais on peut toujours parler de *division euclidienne*. Soient a, b deux entiers naturels, le résultat de la division euclidienne est le couple (q, r) d'entiers naturels tels que $a = b \times q + r$ et $0 \leq r < b$. On dit que q est le *quotient* de la division, et que r est le *reste*.

Nombres entiers relatifs. Le premier défaut des nombres naturels est que la différence n'est pas définir lorsque le premier nombre est supérieur ou égal au second. Pour y remédier, on associe à tout nombre entier naturel non nul n un nombre *opposé*, noté $-n$.

On note $\mathbb{Z}$ l'ensemble $\mathbb{N} \cup \{-n \ ; \ n \in \mathbb{N}^*\}$. Ses éléments sont appelés *entiers relatifs*. On note $\mathbb{Z}^* = \mathbb{Z} - \{0\}$.

Ainsi, $\mathbb{Z} = \{0, 1, -1, 2, -2, 3, -3, \dots\}$, et on a l'inclusion $\mathbb{N} \subset \mathbb{Z}$. L'addition, la multiplication et la soustraction s'étendent) $\mathbb{Z}$, et sont toutes bien définies pour toutes les paires de nombres. La relation d'ordre $\prec$ des entiers naturels s'étend aux entiels relatifs; Elle est toujours compatible avec l'addition puisque si $a \leq b$ et $c \leq d$, alors $a + c \leq b + d$. Les entiers relatifs positifs ou nuls sont les entiers naturels, c'est-à-dire $\mathbb{N} = \{z \in \mathbb{Z} \mid z \geq 0\}$, et les entiers relatifs strictement négatifs sont le complémentaire des entiers naturels, c'est-à-dire $\{z \in \mathbb{Z} \mid z < 0\} = \mathbb{Z} - \mathbb{N}\}$.

En revanche, la relation d'ordre se comporte moins bien vis-à-vis de la multiplication, il faut veiller aux questions de signe : le produit de deux nombres de même signe est positif et le produit de deux nombres de deux signes opposés est négatif.

Un *intervalle* de $\mathbb{Z}$ est un ensemble de la forme $\{z \in \mathbb{Z}\ m \leq z \leq n\}$ pour deux entiers relatifs m et n. On le note alors

$$[\![m, n]\!] = \{z \in \mathbb{Z}\ m \leq z \leq n\}$$

Par exemple, on a $[\![-3, 2]\!] = \{-3, -2, -1, 0, 1, 2\}$. Avec cette notation $[\![m, n]\!]$, on suppose en général que $m \leq n$, mais si jamais on a $m > n$, on a $[\![m, n]\!] = \emptyset$.

Nombres rationnels. Un *nombre rationnel* est un quotient de deux nombres relatifs, le second étant non nul. C'est donc un nombre de la forme $\frac{p}{q}$ avec $p \in \mathbb{Z}$, $q \in \mathbb{Z}^*$. Deux nombres rationnels $\frac{p_1}{q_1}$, $\frac{p_2}{q_2}$ sont déclarés égaux si et seulement si on a $p_1 q_2 = q_1 p_2$.

On dit qu'un nombre rationnel $\frac{p}{q}$ est strictement positif si p et q sont non nuls et de même signe, et strictement négatif si p et q sont non nuls et de signe opposés. Rappelons que l'addition et la multiplication sont bien définies dans $\mathbb{Q}$, par les formules ci-dessous :

$$\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1 q_2 + p_2 q_1}{q_1 q_2}, \qquad \frac{p_1}{q_1} \times \frac{p_2}{q_2} = \frac{p_1 p_2}{q_1 q_2}.$$

Dans $\mathbb{Q}$, tout nombre non nul $\frac{p}{q}$ a un *opposé* pour l'addition, à savoir $\frac{-p}{q}$, et un *inverse* pour la multiplication, à savoir $\frac{q}{p}$. La relation d'ordre $\prec$ des entiers relatifs s'étend aux nombres rationnels en un ordre total encore.

Nombre réels. La définition même de nombre réel est une définition en réalité assez compliquée, sur laquelle beaucoup de visions s'opposent. En réalité, on peut retenir qu'un nombre réel est un nombre pouvant être représenté par une suite de chiffres avec virgule (finie avant la virgule, finie ou infinie après la virgule) et un signe; et réciproquement, toute suite de chiffres finie avant la virgule et finie ou infinie après la virgule équipée d'un signe donne un nombre réel. Ces nombres forment un ensemble, noté $\mathbb{R}$, sur lequel il y a 4 opérations : addition, multiplication, soustraction, division.

Soit x un nombre réel, on appelle *partie entière* de x le nombre entier noté $\lfloor x \rfloor$, qui est le plus grand nombre entier relatif qui est inférieur à x. On appelle la *valeur absolue* de x le nombre réel noté $|x|$ défini par

$$|x| = \begin{cases} x \text{ si } x \geq 0 \\ -x \text{ si } x < 0 \end{cases}$$

Ainsi, on a par exemple $\lfloor \pi \rfloor = 3$, $\lfloor -\pi \rfloor = -4$, $|\pi| = \pi$ et $|-\pi| = \pi$.

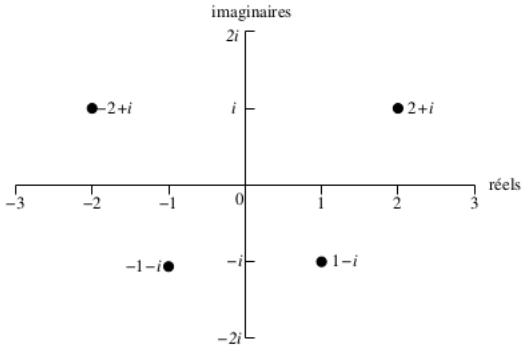
Un *intervalle* de $\mathbb{R}$ est un sous-ensemble J de $\mathbb{R}$ satisfaisant la propriété suivante : si x, y sont deux éléments de J satisfaisant $x \leq y$, alors tous les nombres réels z tels que $x \leq z \leq y$ sont dans J. Un résultat fondamental est que tous les intervalles de $\mathbb{R}$ sont de l'un des types suivants (pour $a, b \in \mathbb{R}$) :

- $[a, b] = \{x \in \mathbb{R}, \mid a \leq x \leq b\}$,
- $[a, b[= \{x \in \mathbb{R}, \mid a \leq x < b\}$,
- $]a, b] = \{x \in \mathbb{R}, \mid a < x \leq b\}$,
- $]a, b[= \{x \in \mathbb{R}, \mid a < x < b\}$,
- $[a, +\infty[= \{x \in \mathbb{R}, \mid a \leq x\}$,
- $]a, +\infty[= \{x \in \mathbb{R}, \mid a < x\}$,
- $]-\infty, b] = \{x \in \mathbb{R}, \mid x \leq b\}$,
- $]-\infty, b[= \{x \in \mathbb{R}, \mid x < b\}$,
- $]-\infty, +\infty[= \mathbb{R}$.

Nombre complexes. Les nombres complexes sont nés de la nécessité de donner un sens à la racine carrée de nombres négatifs pour résoudre des équations. On définit ainsi un nombre i, nommé *nombre imaginaire pur*, qui satisfait $i^2 = -1$. On appelle *nombre complexe* tout nombre de la forme $z = a + ib$ où a et b sont deux réels quelconques. Le réel a est appelé *partie réelle* de z, notée $\mathrm{Re}(z)$, et le nombre b est appelé *partie imaginaire* de z, notée $\mathrm{Im}(z)$. L'ensemble des nombres complexes est noté $\mathbb{C}$. L'addition et la multiplication des réels s'étendent aux nombres complexes comme suit :

1. $(a+ib)+(c+id)=(a+c)+i(b+d)$,
2. $(a+ib)(c+id)=(ac-bd)+i(bc+ad)$.

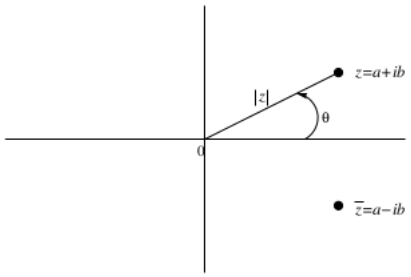
On représente ces nombres par les points d'un plan muni de deux axes orthogonaux. L'axe horizontal porte les réels (qui sont les nombres complexes dont la partie imaginaire est nulle). L'axe vertical porte les nombres dits imaginaires purs, dont la partie réelle est nulle. Le point correspondant au nombre $a+ib$ est placé à la verticale du réel a et à l'horizontale de l'imaginaire pur ib. On dit que le nombre $a+ib$ est l'*affixe* du point qui le représente. Le point d'affixe 0 est l'origine.



Pour un nombre complexe $z=a+ib$, on appelle

1. *modulo* de z le nombre réel positif ou nul $\sqrt{a^2+b^2}$. On le note $|z|$, il représente la distance du point d'affixe z à l'origine du repère.
2. *argument* de z (si z est non nul) un réel θ tel que $\mathrm{Re}(z)=|z|\cos(\theta)$ et $\mathrm{Im}(z)=|z|\sin(\theta)$. On le note $\arg(z)$, et il est défini à 2π ($180°$) près.
3. *décomposition polaire* de z l'écriture $z=|z|e^{i\theta}$, qui est une autre manière très classique de représenter un nombre complexe.
4. *conjugué de* z le nombre complexe

$$\overline{z}:=a-ib.$$



1.1.9. Polynômes. On s'intéresse ici à un autre ensemble contenant des objets mathématiques importants : les polynômes. Les polynômes vont être définis à partir de coefficients qui vont appartenir à l'un des ensembles de nombres définis ci-dessus. Dans la suite, on supposera que ces coefficients sont dans $\mathbb{R}$, mais tout peut s'adapter.

Étant donné un nombre réel x et un nombre entier naturel $n \in \mathbb{N}$, on note x^n le produit de n copies de x. Ainsi, $x^n = x \times x \times \cdots \times x$ avec n fois x et $n-1$ signes $\times$. Par convention, on pose $x^0 = 1$, et on a alors $x^d \times x^{d'} = x^{d+d'}$.

1.1.10 Definition. Étant donné un entier naturel $n \in \mathbb{N}$ et des éléments $a_0, a_1, \ldots, a_n \in \mathbb{R}$, la fonction P qui à un nombre réel x associe le nombre

$$P(x) := a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

est appelé *polynôme* à coefficients dans $\mathbb{R}$. De manière abusive, on parlera de polynôme en x. Si le dernier coefficient a_n est différent de 0, le nombre d est appelé *degré* du polynôme P, noté $\deg(P)$. Dans ce cas, le nombre a_d est appelé *coefficient dominant* de P, noté $\mathrm{dom}(P)$. Si tous les coefficients sont nuls, alors P est appelé *polynôme nul*. Un terme de la forme $a_k x^k$ est appelé un *monôme*.

Étant donné un polynôme P à coefficients dans $\mathbb{R}$, on appelle *racine* de P un nombre $a \in \mathbb{R}$ tel que $P(a) = 0$. Par exemple, $x^{-}1$ est un polynôme en x à coefficients réels qui a deux racines, 1 et -1. On peut additionner, soustraire, ou multiplier deux polynômes. L'addition et la soustraction se font monôme par monôme, par exemple :

$$(x^3 + x^2 + x + 1) + (x^3 + 2x^2 - x - 2) = 2x^3 + 3x^2 - 1,$$

et la multiplication se fait en *développant*, et en regroupant les termes par degré, par exemple :

$$\begin{aligned}(3x^2 + x - 1) \times (x^2 + 2x - 1) &= 3x^2 \times (x^2 + 2x - 1) + x \times (x^2 + 2x - 1) - 1 \times (x^2 + 2x - 1) \\ &= (3x^4 + 6x^3 - 3x^2) + (x^3 + 2x^2 - x) - (x^2 + 2x - 1) \\ &= 3x^4 + (6+1)x^3 + (-3+2-1)x^2 + (-1-2)x - (-1) \\ &= 3x^4 + 7x^3 - 2x^2 - 3x + 1\end{aligned}$$

On peut également équiper l'ensemble des polynômes d'une opération de *division euclidienne* : si A et B sont des polynômes à coefficients réels, alors il existe deux uniques polynômes Q et R tels que

$$A = BQ + R \quad \text{et } \deg(R) < \deg(B).$$

On dit que Q est le *quotient de la division euclidienne* de A par B, et que R est le *reste*. Lorsque R est le polynôme nul, on a $A = BQ$ et on dit alors que B *divise* A, comme ce qui est présenté pour des nombres en Chapitre 2.

1.1.11. Ensembles produits. Pour un entier naturel n, on dit qu'un n-*uplet* est une collection ordonnée de n objets. On le note à l'aide de parenthèse : si $u_1, u_2, \ldots, u_n$ sont des objets, on peut former le n-uplet $(u_1, u_2, \ldots, u_n)$. Deux n-uplets $(u_1, u_2, \ldots, u_n)$ et $(v_1, v_2, \ldots, v_n)$ sont dits égaux si on a $u_1 = v_1$, $u_2 = v_2$, … et $u_n = v_n$. Un 2-uplet est aussi appelé un *couple*; un 3-uplet est appelé un *triplet*.

Attention, un n-uplet $(u_1, u_2, \ldots, u_n)$ n'est pas le même objet qu'un ensemble $\{u_1, u_2, \ldots, u_n\}$. La différence tient au fait que l'ordre des éléments compte dans un n-uplet, et non dans un ensemble. Par exemple, on a $(1,2) \neq (2,1)$, tandis que $\{1,2\} = \{2,1\}$. De plus, un n-uplet peut contenir deux fois le même élément. Ainsi, $(1,1,3)$ est un triplet différent du couple $(1,3)$.

1.1.12 Definition. Étant donnés deux ensembles E et F, l'*ensemble produit* $E \times F$ est l'ensemble dont les éléments sont les couples de la forme (e, f) où e parcourt tous les éléments de E et f tous les éléments de F, c'est-à-dire

$$E \times F = \{(e, f); e \in E, f \in F\}.$$

Pour un entier naturel n, étant donnés n ensembles $E_1, E_2, \ldots, E_n$, l'*ensemble produit* $E_1 \times E_2 \times \cdots \times E_n$ est l'ensemble dont tous les éléments sont les n-uplets de la forme $(u_1, u_2, \ldots, u_n)$ où u_i parcourt tous les éléments de E_i, c'est-à-dire

$$E_1 \times E_2 \times \ldots E_n = \{(u_1, u_2, \ldots, u_n); u_1 \in E_1, u_2 \in E_2, \ldots, u_n \in E_n\}$$

Par exemple, on a

$$\{1, 2, 3\} \times \{3, 4\} = \{(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)\}.$$

1.2. FONCTIONS

Dans cette Section, on s'intéresse à la définition de fonction au sens mathématique. En informatique, une fonction est une portion de code informatique nommée, qui accomplit une tâche spécifique. Les fonctions reçoivent généralement des données en entrée et retournent généralement en sortie le résultat du traitement opéré par la fonction.

1.2.1 Definition. Soient E et F des ensembles, une fonction f de E dans F (aussi appelée *application* de E dans F est définie par son *graphe* : c'est un sous-ensemble Γ_f de $E \times F$ tel que pour tout $x \in E$, exactement un élément y de F vérifie $(x, y) \in \Gamma_f$. Cet élément y est l'image de x et est noté $f(x)$. On rappelle que cet élément est unique ! La notation standard est la suivante :

$$\begin{array}{rccc} f & : & E & \to & F \\ & & x & \mapsto & f(x) \end{array}$$

L'ensemble E est appelé *ensemble de départ* de la fonction f et F est appelé *ensemble d'arrivée*. Lorsque la fonction f est donnée sans précision, on appelle *domaine de définition*, noté $\mathcal{D}_f$, le plus grand sous-ensemble de $\mathbb{Z}$, $\mathbb{R}$ ou $\mathbb{C}$ (selon le contexte) sur lequel f est définie.

En fait, le graphe d'une fonction est l'ensemble des couples $(x, f(x))$ où x parcourt le domaine de définition $\mathcal{D}_f$. On retrouve donc ainsi la description du graphe d'une fonction :

$$\Gamma_f = \{(x, f(x)) ; x \in \mathcal{D}_f\}$$

1.2.2 Remarque. Parfois, on considère une notion un peu plus faible en appelant fonction une application qui n'est pas définie partout : ainsi une fonction associe à un élément de l'ensemble de départ zéro ou une valeur de l'ensemble d'arrivée. L'ensemble de définition d'une telle fonction $f : E \to F$ est un sous-ensemble de E. Cette distinction est parfois pratique, mais nous garderons la notion définie ci-dessus: les fonctions et les applications sont la même chose, et l'image d'un élément de l'ensemble de départ est toujours définie.

Deux fonctions f et g sont dites égales si et seulement si elles ont le même domaine de définition ($\mathcal{D}_f = \mathcal{D}_g$) et si pour tout $x \in \mathcal{D}_f$, $f(x) = g(x)$.

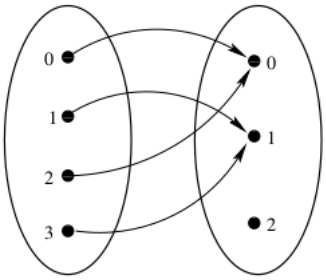
Pour une fonction entre ensembles avec un nombre fini d'éléments, il est commode de représenter le graphe d'une fonction $f : E \to F$ par des flèches entre deux diagrammes de Venn;

pour chaque élément a de E, on trace un point nommé a dans la diagramme correspondant à E et pour chaque élément b de F, on trace un point nommé b dans le diagramme correspondant à F. Pour chaque couple (a, b) de Γ_f, on trace ensuite une flèche allant du point a dans E vers le point b dans F.

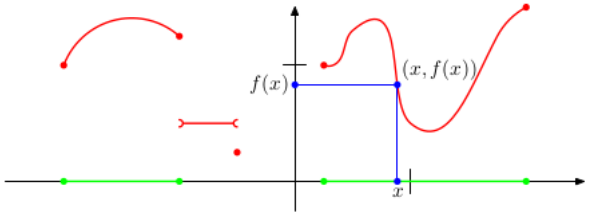
1.2.3 Exemple. Par exemple, considérons $E = \{0, 1, 2, 3\}$ et $F = \{0, 1, 2\}$. Considérons l'application qui à un nombre de E associe le reste de sa division euclidienne par 2, c'est-à-dire 0 s'il est pair, 1 s'il est impair. Le graphe de cette application est

$$\Gamma = \{(0, 0), (1, 1), (2, 0), (3, 1)\},$$

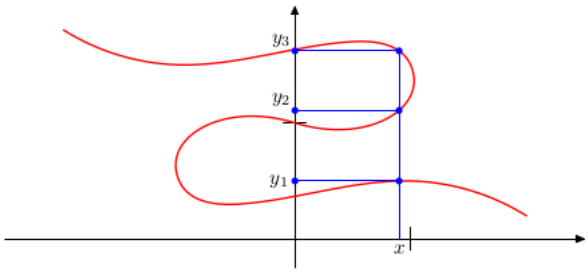
et il est représenté par :



Pour des fonctions d'un sous-ensemble de $\mathbb{R}$ vers $\mathbb{R}$, on représente le graphe dans le plan en dessinant pour chaque élément (x, y) du graphe le point de coordonnées (x, y), voir par exemple ci-dessous. On constate que pour tout x fixé de l'ensemble de définition (en vert), la droite verticale correspondant aux points dont la première coordonnée est x croise le graphe, tracé en rouge, en un unique point de coordonnées $(x, f(x))$. L'*image* $f(x)$ de x est bien la seconde coordonnée de ce point. Notez que si pour un x donné, la droite verticale croise plusieurs fois le graphe, alors on ne trace pas une fonction puisque tout élément n'a pas une seule image.



Est une fonction



N'est pas une fonction

1.2.4. Suites. La notion de suite, vue dans le secondaire, est en fait un cas particulier de la définition de fonction.

Une *suite* d'éléments d'un ensemble E est une fonction $u : \mathbb{N} \to E$. Au lieu d'utiliser la notation fonctionnelle classique, on préfèrera la notation indicielle

$$u_n := u(n), \quad \text{pour } n \in \mathbb{N}.$$

On parlera alors de suite $(u_n)_{n\in\mathbb{N}}$. Par abus de langage, si $n_0 \in \mathbb{N}$, on appellera également de suite d'éléments de E une fonction de $\{n \in \mathbb{N} \mid n \geq n_0\}$ dans E, que l'on notera $(u_n)_{n\geq n_0}$. On parlera alors de *suite définie à partir d'un certain rang*.

1.2.5 Exemple. Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie pour $n \in \mathbb{N}$ par $u_n = (-1)^n$. On a $u_0 = 1$, $u_1 = -1$, etc.

On peut aussi définir une suite par une *relation de récurrence*. Par exemple, soit $(u_n)_{n\in\mathbb{N}}$ une suite définie par $u_0 = 0$ et pour tout $n \geq 0$ par $u_{n+1} = 3u_n + 4$. On a alors $u_1 = 4$, $u_2 = 16$, etc.

Une suite définie par une relation de récurrence est à rapprocher d'une fonction *récursive*. En effet, le premier terme de la suite fournit le cas d'arrêt de la fonction, tandis que la formule de récurrence fournit l'appel récursif.

1.2.6 Exercice. La suite de Fibonacci est définie par

$$u_0 = 1, u_1 = 1, \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n.$$

Écrire en Python 3 fonctions permettant de calculer le nombre u_n pour un entier n passé en paramètre :

1. une fonction récursive,
2. une fonction récursive terminale,
3. une fonction itérative.

1.2.7. Image et préimage d'un ensemble. Soient **E** et **F** deux ensembles, et $f : \mathbf{E} \to \mathbf{F}$ une fonction.

(1) Soit A un sous-ensemble de **E**. On appelle *image* de A par f, que l'on note $f(A)$, l'ensemble
$$f(A) = \{f(x) \mid x \in A$$
contenant l'ensemble des images des éléments de A.

(2) Soit B un sous-ensemble de **F**. On appelle *image réciproque*, ou *préimage* de B par f, que l'on note $f^{-1}(B)$ l'ensemble des éléments de **E** dont l'image appartient à B :
$$f^{-1}(B) = \{x \in \mathbf{E} \mid f(x) \in B\}$$

Dans la définition de préimage, attention à la notation f^{-1}, dont nous verrons une autre version par la suite. Elle ne désigne pas que f est inversée.

1.2.8 Exemple. Dans l'application $\{0,1,2,3\} \to \{0,1,2\}$ ci-dessus, on a :

1. $f(\{0,2\}) = \{0\}$,
2. $f(\{1\}) = \{1\}$,
3. $f^{-1}(\{1\}) = \{1,3\}$,
4. $f^{-1}(\{2\}) = \emptyset$.

Un élément x de E tel que $f(x) = y$ s'appelle un *antécédent de* y. D'après la définition précédente, $f^{-1}(\{y\})$ correspond à l'ensemble des antécédents de $y \in \mathbf{F}$. Voici ci-dessous les interprétations graphiques des notions d'image directe et d'image réciproque, pour des fonctions représentées par des diagrammes de Venn et des graphes :

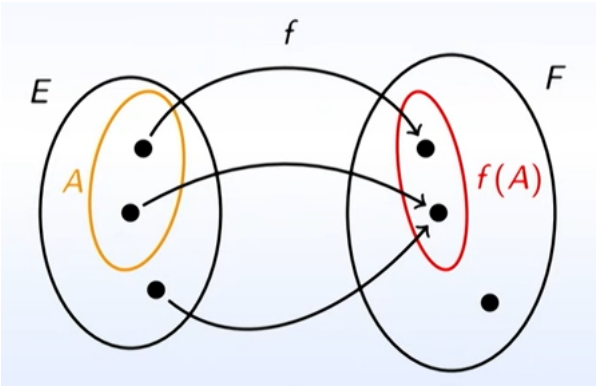


Image directe

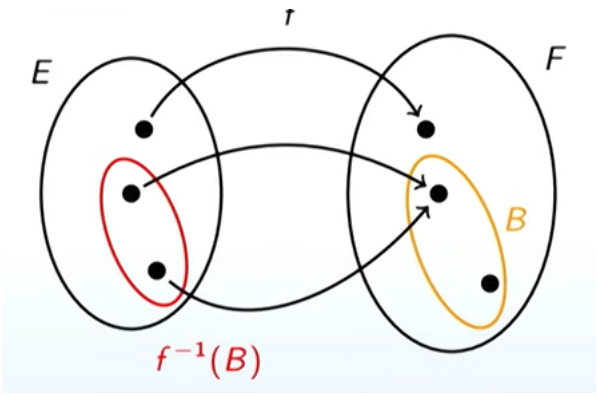


Image réciproque

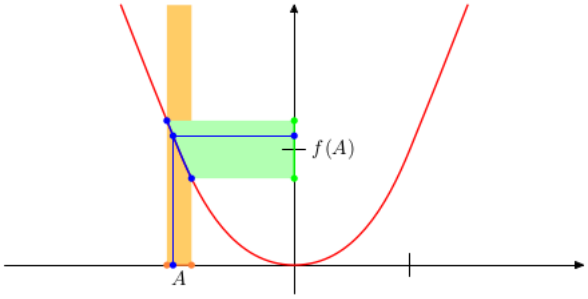


Image directe

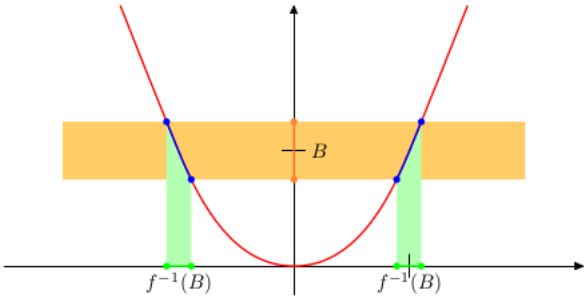


Image réciproque

1.2.9. Composition de fonctions. Soient E, F et G des ensembles, $f : E \to F$ et $g : F \to G$ des fonctions. On définit la *composée* de f par g, notée $g \circ f$, comme la fonction $E \to G$ qui à $x \in E$ associe

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

Attention à l'ordre des applications dans l'écriture $g \circ f$, c'est une notation **fonctionnelle**, habituelle dans les langages de programmation fonctionnels. C'est l'ordre inverse des flèches dans le schèma ci-dessous.

$$\begin{array}{ccccc} E & \xrightarrow{f} & F & \xrightarrow{g} & G \\ x & \mapsto & f(x) & \mapsto & g(f(x)) \end{array}$$

1.2.10. Injectivité, surjectivité, bijectivité. Soient **E**, **F** des ensembles et $f : \mathbf{E} \to \mathbf{F}$ une fonction. On dit que f est :

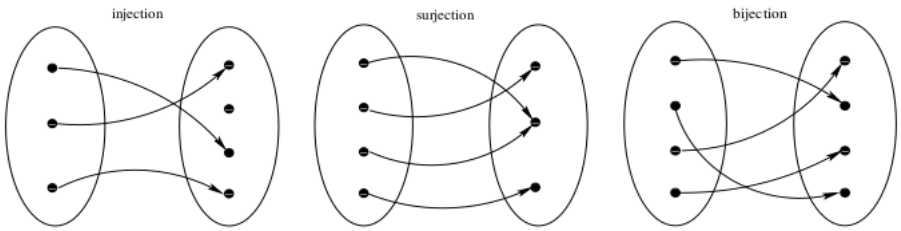
1. *injective* si tout élément de l'ensemble d'arrivée **F** possède **au plus** un antécédent dans l'ensemble de départ **E** :

$$\forall x_1, x_2 \in \mathbf{E}, ((f(x_1) = f(x_2)) \Rightarrow x_1 = x_2.$$

2. *surjective* si tout élément de l'ensemble d'arrivée **F** possède **au moins** un antécédent dans l'ensemble de départ **E** :

$$\forall y \in \mathbf{F}, \ \exists x \in \mathbf{E}, \ f(x) = y.$$

3. *bijective* si tout élément de l'ensemble d'arrivée **F** possède **exactement** un antécédent dans l'ensemble de départ **E**. Une fonction bijective est donc une fonction qui est à la fois injective et surjective.



1.2.11 Remarque. Pour une fonction $f : \mathbf{E} \to \mathbf{F}$ entre ensembles ayant un nombre fini d'éléments, on a les faits suivants :

1. Si f est injective, alors il y a plus d'éléments dans l'ensemble d'arrivée **F** que dans l'ensemble de départ **E**.

2. Si f est surjective, alors il y a moins d'éléments dans l'ensemble d'arrivée **F** que dans l'ensemble de départ **E**.

3. Si f est bijective, alors il y a exactement le même nombre d'éléments dans l'ensemble d'arrivée **F** et dans l'ensemble de départ **E**.

1.2.12. Fonction réciproque. Si une fonction $f : \mathbf{E} \to \mathbf{F}$ est **bijective**, tout élément y de **F** a un et un seul antécédent, que l'on note $f^{-1}(y)$. On peut alors définir la *fonction réciproque* de f, notée f^{-1}, dont l'ensemble de départ est **F** et d'arrivée est **E**, par

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

Ainsi, si f est bijective, la composée de f par son application réciproque f^{-1} est la fonction $E \to E$, qui à x associe x. On l'appelle application identité. On a :

$$\begin{array}{ccccc} \mathbf{E} & \xrightarrow{f} & \mathbf{F} & \xrightarrow{g} & \mathbf{E} \\ x & \mapsto & f(x) & \mapsto & f^{-1}(f(x)) = x. \end{array}$$

Notons que les notations pour la fonction réciproque et et l'image réciproque d'un sous-ensemble B de F sont liées par la relation

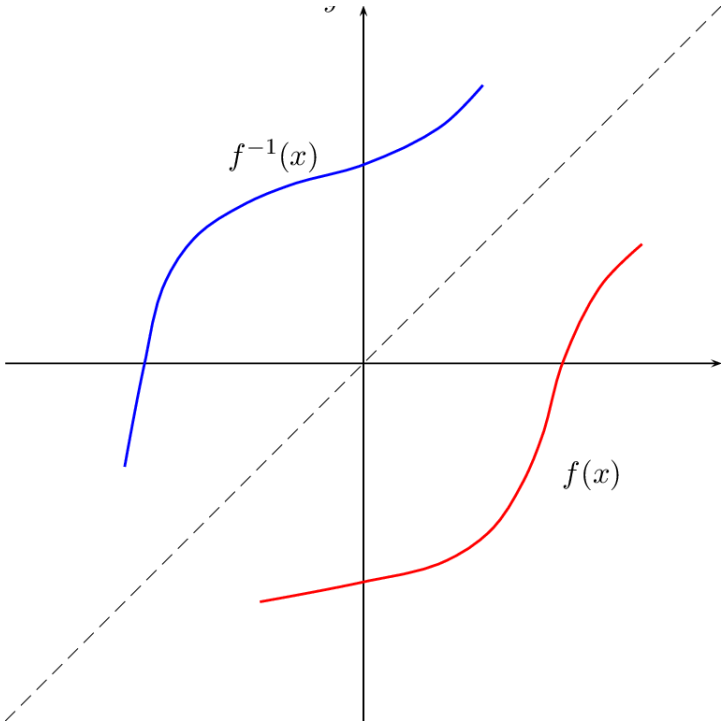
$$f^{-1}(\{y\}) = \{f^{-1}(y)\}.$$

1.2.13 Exemple. Les fonctions *logarithme néperien* $x \in]0, +\infty[\mapsto \ln(x)$ et *exponentielle* $x \in \mathbb{R} \mapsto e^x$ sont réciproques l'une de l'autre, on a donc

$$\forall x \in \mathbb{R}, \ln(e^x) = x, \quad \forall y \in]0, +\infty[, e^{\ln(y)} = y.$$

En terme de représentation graphique, le graphe de deux fonctions réciproques l'une de l'autre sont symétriques par rapport à la bissectrice du plan, c'est-à-dire la droite d'équation

$y = x$.



1.3. Logique propositionnelle

1.3.1. Assertions, propositions logiques. En informatique, et notamment en programmation, il est souvent question de boucles conditionnelles, dans lesquelles le programmeur cherche à déterminer si une condition est vraie afin d'exécuter telle ou telle série d'instructions. Une telle condition est ce qu'on appelle sur le plan mathématique une *assertion*, c'est à dire une affirmation mettant en jeu des objets mathématiques.

Quelques exemples:

- 2 est un nombre pair;
- -5 est un nombre réel positif;
- 3 est égal à 5;
- Il existe une infinité de nombres premiers p tels que $p + 2$ est premier.

Les assertions ont en général une *valeur de vérité*, c'est-à-dire que l'on peut en général déterminer si elles sont vraies ou fausses. Dans les exemples précédents, on peut facilement dire que la première est vraie, la deuxième et la troisième sont fausses, et en réalité nul ne sait aujourd'hui si la dernière est vraie ou fausse. Notez également qu'une assertion peut se traduire de plusieurs manières différentes: en informatique, la première assertion sera plutôt écrite sous la forme

- La division de 2 par 2 est un nombre entier;
- Le reste dans la division euclidienne de 2 par 2 est 0;
- 2 modulo 2 est égal à 0.

Nous verrons par la suite comment donner un sens à la notion d'équivalence d'assertions.

A titre d'information, on utilise souvent d'autres termes pour désigner certains types d'assertions mathématiques, pour marquer leur importance ou leur difficulté:

- *Théorème :* c'est une assertion importante, que l'on peut démontrer.
- *Proposition :* c'est le terme qui est utilisé le plus souvent, pour désigner une assertion qui est vraie, que l'on sait démontrer mais qui n'est pas aussi importante et plus facile à démontrer qu'un théorème.
- *Lemme :* c'est une assertion démontrée, qui constitue une étape dans la démonstration d'un théorème.
- *Corollaire :* c'est une conséquence facile d'un théorème.
- *Axiome :* c'est une assertion dont on admet qu'elle est vraie. On parle aussi de *postulat*. Ces assertions sont les fondations de l'édifice mathématique, sans lesquelles on ne pourrait démontrer grand chose.
- *Conjecture :* c'est une assertion dont certains mathématiciens pensent qu'elle est vraie, mais personne ne l'a encore démontrée.

En revanche, l'assertion « x est supérieur à 3 » n'a pas de valeur de vérité en tant que tel, étant donné que l'on ne dispose pas d'information sur x, ni même sur sur son *type*, c'est-à-dire l'ensemble auquel il appartient. Selon la valeur de x, cette affirmation sera soit vraie, soit fausse. Ici apparaît un concept important, celui de *variable*.

1.3.2 Definition. Une variable est un symbole représentant un objet mathématique dont la valeur n'est pas connue. Lorsqu'une variable est introduite dans un énoncé en disant à quel ensemble elle appartient, on parle de variable liée, ou typée. Sinon, elle est dite libre ou non typée.

Une assertion ne contenant que des variables liées est dite *close*. Une telle assertion est soit vraie, soit fausse.

Une assertion contenant au moins une variable libre est dite *ouverte*. On dit qu'elle dépend des paramètres que sont les variables libres, et une telle assertion n'est donc ni vraie ni fausse. En revanche, lorsque les paramètres sont fixés, elle hérite également d'une valeur de vérité.

Exemples:

- « x est supérieur à 3 » est une assertion ouverte, où x est une variable libre
- « Pour tout nombre réel x, x est supérieur à 3 » est une assertion close, qui est fausse. En effet, si on prend par exemple pour x la valeur 0, alors il est faux de dire que 0 est supérieur à 3.
- « Pour tout nombre réel x, si x est supérieur à 5, alors x est supérieur à 3 » est une assertion close, qui est vraie.

1.3.3 Definition. En logique classique, une *proposition* est une affirmation formée d'assertions (ouvertes ou closes) en nombre fini arbitraire, connectées entre elles par ce qu'on appelle des *opérateurs logiques* ou *connecteurs logiques*. Ces opérateurs sont:

- la négation « non », notée $\neg$;
- la conjonction « ou », notée $\wedge$;
- la conjonction « et », notée $\vee$;
- l'implication « si, alors », notée $\Rightarrow$;
- l'équivalence, ou bi-implication « si et seulement si », notée $\Leftrightarrow$.

Ainsi, une proposition est formée d'un nombre quelconque n d'assertions reliées entre elles par ces connecteurs. Une assertion close est donc en elle-même déjà une proposition; nous utiliserons donc maintenant le terme proposition. Nous les noterons par les lettres $P, Q, R, \ldots$

Voici quelques propositions sur un nombre n entier, et leur traduction en français:

- $\neg(n < 5)$: « n n'est pas strictement inférieur à 5 »;
- $(n < 5) \wedge (n \leq 2)$: « n est strictement inférieur à 5 et inférieur ou égal à 2 »;
- $(n \geq 2) \vee (n \leq 0)$: « n est supérieur ou égal à 2 ou inférieur ou égal à 0 »
- $(n \leq 5) \Rightarrow (n < 4)$: « n est inférieur ou égal à 5 implique que n est strictement inférieur à 4 » ou encore « Si n est inférieur ou égal à 5, alors n est strictement inférieur à 4. »

1.3.4. Tables de vérité. Puisqu'une proposition est par principe soit vraie, soit fausse, on lui attribuera une valeur de vérité « vrai » ou « faux », ce que l'on représentera par 1 ou 0 respectivement. On peut décider de la valeur de vérité d'une proposition en fonction de la valeur de vérité des assertions qui la composent à l'aide d'une **table de vérité**.

Commençons par donner les tables de vérité associées aux propositions simples formées par une ou deux assertions, et les opérateurs logiques $\neg$, $\wedge$, $\vee$:

Table de vérité de la négation $\neg$:

P	$\neg P$
0	1
1	0

Cette table peut se lire de la manière suivante: si la proposition P est fausse, alors $\neg P$ est vraie (Ligne 1); et si la proposition P est vraie, alors $\neg P$ est fausse (Ligne 2).

Table de vérité de la conjonction $\wedge$:

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité de la disjonction $\vee$:

P	Q	$P \vee Q$
0	0	0
0	1	1
1	0	1
1	1	1

Notons ici que le « ou » de la langue française est ambigu. En général, il correspond à un « ou » *exclusif*, qui sera vrai si l'une des deux assertions qui le composent est vrai, mais pas les deux. Par exemple, dans la phrase « Fromage ou dessert », on entend que l'on peut choisir l'un ou l'autre, mais pas les deux. Le « ou » de la conjonction correspond à un « ou » dit *non-exclusif*, c'est-à-dire que la proposition est vraie à partir du moment où l'une des deux assertions est vraie, y compris les deux. Cela distingue du « ou » exclusif, en général noté $\oplus$, dont la table de vérité est donnée par

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité de l'implication :

P	Q	$P \Rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

Il faut ici retenir que quelque chose de faux impliquera toujours du vrai. Lorsque P est vraie, si Q est faux, alors $P \Rightarrow Q$ est fausse, par exemple « 2 est pair implique $0 = 1$ » est une assertion fausse. Si Q est vraie, alors $P \Rightarrow Q$ est vraie.

1.3.5 Remarque. On peut noter que si une proposition logique P est composée de n assertions, alors la table de vérité associée à P contiendra 2^n lignes : pour la remplir il faudra lister tous les cas possibles 0 ou 1 pour chacune de ces n assertions.

1.3.6 Definition. Soient P et Q deux propositions logiques, dépendant éventuellement de variables libres. On dit que P et Q sont *équivalentes*, et on note $P \equiv Q$ si elles ont la même valeur de vérité, c'est-à-dire si elles sont toutes les deux vraies pour exactement les mêmes paramètres.

Autrement dit, P et Q sont équivalentes si, lorsqu'on construit une table de vérité pour P et une table de vérité pour Q, tous les résultats des 2 colonnes sont identiques.

Par exemple, si P et Q sont deux assertions, les propositions $P \Rightarrow Q$ et $(\neg P) \vee Q$ sont équivalentes, comme démontré ci-dessous:

P	Q	$P \Rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

P	Q	$\neg P$	$(\neg P) \vee Q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

D'ailleurs, c'est en général la définition qui est donnée pour la proposition $P \Leftrightarrow Q$, les trois opérateurs $\neg, \wedge, \vee$ étant considérés comme les trois essentiels uniquement.

La proposition $P \Leftrightarrow Q$ est de même équivalente à $(P \Leftarrow Q) \wedge (Q \Leftarrow P)$, d'où le terme de « double implication ». Ainsi, $P \Leftrightarrow Q$ est vraie si à la fois P implique Q et Q implique P.

1.3.7 Definition. Une proposition logique est appelée une *tautologie* si elle est vraie quelle que soit la valeur de vérité des assertions qui la composent. Autrement dit, sa table de vérité ne contient que des 1.

Par exemple, la proposition $P \vee (\neg P)$ est une tautologie:

P	$\neg P$	$P \vee (\neg P)$
0	1	1
1	0	1

Exercice. Pour des assertions P, Q, R, les propositions suivantes sont-elles des tautologies ?

- $((P \Rightarrow Q) \wedge R) \Rightarrow Q$;
- $(P \wedge Q) \vee \neg(P \wedge R)$;
- $\neg P \Rightarrow (P \Rightarrow Q)$.

1.3.8. Règles de négation. On dispose de règles simples permettant de considérer la négation d'une assertion formée des trois opérateurs $\neg, \wedge$ et $\vee$. Pour des assertions P, Q, R, on a les équivalences entre les propositions suivantes:

- $\neg(\neg P) \equiv P$;
- $\neg(P \vee Q) \equiv (\neg P) \wedge (\neg Q)$;
- $\neg(P \wedge Q) \equiv (\neg P) \vee (\neg Q)$;
- $\neg(P \Rightarrow Q) \equiv P \wedge (\neg Q)$.

1.3.9. Quantificateurs. Les quantificateurs sont des objets primordiaux en mathématiques : ils permettent de choisir un élément ou plusieurs dans un ensemble. Il y en a deux, qui dépendent du fait que l'on prenne tous les éléments ou que l'on en choisisse un.

1.3.10 Definition. Les quantificateurs sont les deux symboles $\forall$, qui se traduit par « quel que soit » et $\exists$, qui se traduit par « il existe ». Si $P(x)$ est une assertion dépendant d'une variable x, alors

- l'assertion $(\forall x \in E, P(x))$ est vraie si et seulement si l'assertion $P(x)$ est vraie pour n'importe quel x dans E.
- l'assertion $(\exists x \in E, P(x))$ est vraie si et seulement si l'assertion $P(x)$ est vraie pour au moins un x dans E.

Par exemple, l'assertion

$$\forall n \in \mathbb{N}, \exists m \in \mathbb{N},\ n < m$$

se lit

> « Pour tout entier naturel n, il existe un entier naturel m tel que n est strictement inférieur à m. »

Il faut bien retenir que dans ce cas, m peut dépendre de l'entier n. Ainsi, on peut constater que cette assertion est vraie, puisque si on prend un entier naturel n **quelconque**, on peut bien trouver un entier m supérieur, par exemple $m = n + 1$ est strictement supérieur à n.

1.3.11 Remarque. (1) Le nom ou la lettre qu'on utilise pour désigner une variable n'a pas d'importance en général. On peut la changer, du moment qu'on le change à chaque occurrence de la variable. Ainsi, les deux assertions ci-dessous sont équivalentes:

$$(\forall n \in \mathbb{N}, \exists m \in \mathbb{N},\ n < m) \equiv (\forall x \in \mathbb{N}, \exists y \in \mathbb{N},\ x < y)$$

En revanche, elles ne sont pas équivalentes à

$$(\forall n \in \mathbb{N}, \exists n \in \mathbb{N},\ n < n)$$

qui n'a pas de sens car la variable n est introduit 2 fois dans la même assertion. Comme en informatique, on ne peut pas nommer de la même manière deux variables différentes.

(2) L'ordre dans lequel on écrit les quantificateurs est très important. Si on échange les deux quantificateurs dans la proposition précédente, on obtient

$$(\exists n \in \mathbb{N}, \forall m \in \mathbb{N},\ n < m)$$

qui se traduit par

> « Il existe un entier naturel n tel que, pour tout entier naturel m, n est strictement inférieur à m. »

ce que l'on peut reformuler par

> « Il existe un entier naturel n strictement inférieur à tous les entiers naturels »

qui est une assertion fausse !

1.3.12 Proposition. *Soit* $P(x)$ *une assertion dépendant d'une variable* x*, et soit* E *un ensemble. Alors les assertions :*

- $\neg(\forall x \in E, P(x))$ *et* $(\exists x \in E, \neg P(x))$ *sont équivalentes.*
- $\neg(\exists x \in E, P(x))$ *et* $(\forall x \in E, \neg P(x))$ *sont équivalentes.*

Il faut retenir de ce résultat que pour écrire la négation d'une assertion avec un quantificateur, il faut changer les $\forall$ en $\exists$ et les $\exists$ en $\forall$, puis écrire la négation de l'assertion qui suit la liste de quantificateurs. Ceci est conforme à l'intuition, puisque la négation de « Tous les x de E vérifient $P(x)$ » est « Il existe un x dans E qui ne vérifie pas $P(x)$. »

Pour les deux autres opérateurs logiques, il faut se méfier, il y a deux cas où l'on peut distribuer, et deux cas où l'on ne peut pas: soient $P(x)$ et $Q(x)$ des assertions dépendant d'une variable x, et soit E un ensemble. Les assertions

- $\exists x \in E, P(x) \vee Q(x)$ et $(\exists x \in E, P(x)) \vee (\exists y \in E, Q(y))$ sont équivalentes.
- $\forall x \in A, P(x) \wedge Q(x)$ et $(\forall x \in E, P(x)) \wedge (\forall y \in E, Q(y))$ sont équivalentes.

Il faut être vigilant dans l'utilisation de telles propriétés. Le quantificateur $\exists$ n'est pas distributif par rapport à $\wedge$, tout comme le quantificateur $\forall$ n'est pas distributif par rapport à $\vee$. En effet, pour illustrer ceci, notons que la proposition

> « Il existe un entier supérieur à 7 et inférieur à 6 »

est fausse, tandis que la proposition

> « Il existe un entier supérieur à 7 et il existe un entier inférieur à 6 »

est vraie.

1.3.13. Satisfaisabilité, tables de Karnaugh. Une proposition logique est dite *satisfaisable* si la dernière colonne de sa table de vérité contient au moins un 1; autrement dit, il existe une combinaison des valeurs de vérité des assertions la composant qui la rend vraie.

1.3.14 Definition. Une *clause* est une disjonction de variables propositionnelles ou de négation de variables propositionnelles.

Par exemple, $P \vee Q \vee \neg R$ est une clause. On peut démontrer que toute proposition est équivalente à une conjonction de clauses.

1.3.15 Definition.

- Une *forme normale conjonctive* d'une proposition P est une conjonction de clauses logiquement équivalente à P.
- Une *forme normale disjonctive* d'une proposition P est une disjonction de conjonctions d'assertions ou de négations d'assertions équivalente à P.

Considérons une proposition

$$A := \neg(P \Rightarrow (Q \Rightarrow R \wedge S))) \Leftrightarrow (\neg R \wedge (Q \vee (R \Leftrightarrow P \vee \neg S))))$$

dont nous dressons la table de vérité ci-dessous, ainsi que de sa négation

P	Q	R	S	$\neg(P \Rightarrow (Q \Rightarrow R \wedge S)))$	$(\neg R \wedge (Q \vee (R \Leftrightarrow P \vee \neg S))))$	A	$\neg A$
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	0
1	0	1	0	0	0	1	0
1	0	1	1	0	0	1	0
1	1	0	0	1	1	1	0
1	1	0	1	1	1	1	0
1	1	1	0	1	0	0	1
1	1	1	1	0	0	1	0

Si on regarde la colonne associée à $\neg A$, il n'y a que quatre combinaisons de valeurs de vérité de P, Q, R, S pour lesquelles $\neg A$ est vraie. On en déduit alors une forme normale disjonctive pour $\neg A$, consituée de quatre clauses correspondant aux lignes de la table donnant un 1:

$$\neg A \equiv (\neg P \wedge \neg Q \wedge \neg R \wedge S) \vee (\neg P \wedge Q \wedge \neg R \wedge \neg S) \vee (\neg P \wedge Q \wedge \neg R \wedge S) \vee (P \wedge Q \wedge R \wedge \neg S)$$

et en prenant la négation de cette forme disjonctive, on obtient une forme normale conjonctive pour A:

$$A \equiv (P \vee Q \vee R \vee \neg S) \wedge (P \vee \neg Q \vee R \vee S) \wedge (P \vee \neg Q \vee R \vee \neg S) \wedge (\neg P \vee \neg Q \vee \neg R \vee S)$$

Le problème de satisfaisabilité (ou problème SAT), cf Wikipiedia, est un problème de décidabilité très connu en informatique, à la base de la théorie de la complexité. Sa formulation est la suivante: étant donnée une proposition sous forme normale conjonctive, peut-on déterminée si elle est satisfaisable ou non ?

En fait, la méthode ci-dessus fournit un algorithme pour répondre à cette question, et ce quelle que soit la proposition donnée. Ce problème est donc décidable, mais le problème principal réside dans le fait qu'il s'agit d'un problème difficile, au sens où il n'existe pas d'algorithme rapide pour le résoudre. En effet, pour une proposition à n variables, la table de vérité contient 2^n lignes et ainsi la complexité dans le pire des cas d'une telle méthode est exponentielle. De plus, la forme normale conjonctive est généralement bien plus longue que la proposition de départ, puisqu'elle est construite sur les 1 de la table de vérité, et il peut donc y en avoir 2^n...

On peut alors se demander si il existe un procédé permettant de trouver des propositions équivalentes les plus compactes possibles. C'est ici qu'interviennent les *tables de Karnaugh*. Une table de Karnaug d'une proposition est un tableau à deux dimensions construit comme suit. On sépare les assertions de notre proposition en deux parties:

- si le nombre d'assertions est pair, égal à $2n$, on sépare en deux parties de taille n;
- si le nombre d'assertions est impair, égal à $2n+1$, on sépare en deux parties de taille respective n et $n+1$

Les différentes distributions des valeurs de vérité de chaque partie (sous forme de n-uplets ou n-uplets et $(n+1)$-uplets) sont listés verticalement et horizontalement, de sorte que deux entrées sur deux colonnes ou ligne consécutive ne diffèrent que sur un élément. On construit ainsi un tableau à n^2 ou $n(n+1)$ cases, dans lesquelles on inscrit la valeur de vérité de la proposition correspondant aux valeurs de vérité des assertions données par la ligne et la colonne de cette case. Le but est de faire apparaître visuellement des simplifications possibles. Voici un exemple d'une table de Karnaugh pour la proposition

$$A := \neg(P \Rightarrow (Q \Rightarrow R \wedge S))) \Leftrightarrow (\neg R \wedge (Q \vee (R \Leftrightarrow P \vee \neg S))))$$

On sépare les assertions P, Q, R, S en deux groupes: (P, Q) verticalement et (R, S) horizontalement.

	$(0,0)$	$(0,1)$	$(1,1)$	$(1,0)$
$(0,0)$	1	0	1	1
$(0,1)$	0	0	1	1
$(1,1)$	1	1	1	0
$(1,0)$	1	1	1	1

Trouver une forme normale disjonctive minimale. Dans la table de Karnaugh, on va considérer que la dernière ligne est adjacente en dessous à la première ligne, et que la dernière colonne, sera adjacente à gauche à la première colonne. En gros, les lignes et les colonnes se répètent.

On va regrouper les valeurs de la table égales à 1 en *rectangles* contenant un nombre de cases égales à une puissance de 2, par xemple $1, 2, 4, 8, 16$,etc. Un rectangle de taille $n \times m$ est obtenu en sélectionnant n lignes et m colonnes adjacentes. On doit alors trouver le nombre de rectangles les plus grands possibles, ne contenant bien sûr aucun 0. Un 1 peut appartenir à plusieurs rectangles.

De par la définition des tables de Karnaugh, chaque rectangle correspond à une conjonction d'assertions ou de négation d'assertions (puisque les lignes et colonnes adjacentes ne diffèrent

que d'une composante). La disjonction de ces conjonctions donnera une proposition équivalente à la proposition de départ. Voici la procédure à suivre pour trouver une forme normale disjonctive de longueur minimale:

- On teste tous les rectangles de taille maximale $n \times n$ ou $n \times (n+1)$, puis tant qu'on a un 0 dans le rectangle, on diminue les tailles considérées (en ne gardant que des puissances de 2 !) dans un ordre approprié, jusqu'à examiner les carrés de taille 1.

Dans l'exemple de la table de Karnaugh ci-dessus, on teste alors les rectangles de taille 16,8,4,2 et 1. Aucun rectangle de taille 16 ou 8 ne contient que des 0. En revanche, les quatre rectangles de taille 4 ci-dessous respectent les contraintes:

	$(0,0)$	$(0,1)$	$(1,1)$	$(1,0)$
$(0,0)$	1	0	**1**	**1**
$(0,1)$	0	0	**1**	**1**
$(1,1)$	1	1	1	0
$(1,0)$	1	1	1	1

Dans ce rectangle, on remarque que P est toujours faux, et R est toujours vrai, tandis que les assertions Q et S varient. Il correspondra donc à $(\neg P \wedge R)$.

	$(0,0)$	$(0,1)$	$(1,1)$	$(1,0)$
$(0,0)$	**1**	0	1	**1**
$(0,1)$	0	0	1	1
$(1,1)$	1	1	1	0
$(1,0)$	**1**	1	1	**1**

Ce rectangle 2×2 est valide puisqu'on rappelle que les dernières lignes et colonnes sont adjacentes aux premières. Selon le même principe, il correspond à la proposition $(\neg Q \wedge \neg R)$.

	$(0,0)$	$(0,1)$	$(1,1)$	$(1,0)$
$(0,0)$	1	0	1	1
$(0,1)$	0	0	1	1
$(1,1)$	**1**	**1**	1	0
$(1,0)$	**1**	**1**	1	1

Le rectangle ci-dessus correspond à la proposition $(P \wedge \neg R)$. Et enfin le dernier rectangle, qui lui est de taille 4×1 et non 2×2 est le suivant:

	$(0,0)$	$(0,1)$	$(1,1)$	$(1,0)$
$(0,0)$	1	0	**1**	1
$(0,1)$	0	0	**1**	1
$(1,1)$	1	1	**1**	0
$(1,0)$	1	1	**1**	1

Il correspond à la proposition $(R \wedge S)$.

Ainsi, nous obtenons la forme normale disjonctive équivalente à A de longueur 8 suivante:

$$(\neg P \wedge R) \vee (\neg Q \wedge \neg R) \vee (P \wedge \neg R) \vee (R \wedge S).$$

Notez qu'elle est beaucoup plus courte que la forme normale disjonctive obtenue à partir de la table de vérité, qui est elle de longueur 48, puisque la table de vérité possède 12 fois la valeur 1, chacun de ces 1 étant décrit par une conjonction de longueur 4, qui est le nombre d'assertions. Trouver une forme normale conjonctive minimale.- Pour trouver une forme normale conjonctive minimale d'une proposition A, on va reprendre le principe précédent pour trouver une forme normale disjonctive minimale pour $\neg A$. Il suffit ensuite de prendre sa négation pour trouver la forme normale cherchée, dont on pourrait démontrer qu'elle est encore bien de longueur minimale.

1.3.16. Quelques techniques de démonstration. Dans cette section, nous allons voir plusieurs méthodes de démonstration d'assertions mathématique, basées sur des raisonnements logiques.

Preuve par raisonnement déductif. Le type de preuve le plus classique en mathématique est la démonstration d'une implication $P \Rightarrow Q$. Pour démontrer la véracité d'une telle assertion, on supposera en général que P est vraie, et on essayera d'aboutir à la preuve que Q est vraie. On parle alors de raisonnement *direct*.

Parmi les propriétés fondamentales de l'implication, on peut notamment énoncer les axiomes mathématiques suivant: soit $P(x)$ une proposition dépendant d'une variable x, soit E un ensemble et a un élément de E. Alors les assertions

$$(\forall x \in E, P(x)) \Rightarrow P(a)) \quad \text{et} \quad P(a) \Rightarrow (\exists x \in E, P(x))$$

sont des tautologies. Par exemple, l'assertion « $\forall x \in \mathbb{R}, x^2 - 2x + 1 \geq 0$ » implique l'assertion « $\pi^2 - 2\pi + 1 \geq 0$ ».

1.3.17 Remarque. (1) Ces deux énoncés sont conformes à l'intuition. Le second correspond à la façon standard de démontrer une assertion d'existence : pour démontrer que « $\exists x \in E, P(x)$ », il suffit de trouver un élément $a \in E$ tel qu'on puisse prouver $P(a)$.

(2) Pour démontrer qu'une assertion « $\forall x \in E, P(x)$ » est faussev, il suffit d'exhiber un *contre-exemple*, c'est-à-dire un élément $a \in E$ tel que $\neg P(a)$ est vraie.

(3) Pour démontrer qu'une assertion « $\exists x \in E, P(x)$ » est vraie, on commencera par choisir un x quelconque dans E avec la phrase « Soit $x \in E$. » De cette manière, dans la démonstration qui sit, x désigne un élément arbitraire et fixé de E. Si on démontre $P(x)$, alors comme cela vaut pour x arbitraire on aura démontré la proposition. Par exemple, démontrons l'assertion $\forall t \in \mathbb{R}, t^2 + 2t + 2$.

Preuve : Soit $t \in \mathbb{R}$, on a $t^2 + 2t + 2 = (t + 1)^2 + 1$. Comme le carré d'un nombre réel est positif, $(t + 1)^2 \geq 0$ et donc $(t + 1)^2 + 1 \geq 0$, ce qui prouve notre assertion.

Preuve par contraposée. Ce type de raisonnement est aussi employé pour démontrer une implication $P \Rightarrow Q$. Il consiste à démontrer, au lieu de $P \Rightarrow Q$, la proposition *contraposée* qui est $(\neg Q) \Rightarrow (\neg P)$, qui est équivalente.

P	Q	$P \Rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

P	Q	$\neg Q$	$\neg P$	$(\neg Q) \Rightarrow (\neg P)$
0	0	1	1	1
0	1	0	1	1
1	0	1	0	0
1	1	0	0	1

Par exemple: on veut démontrer la proposition suivante:

« Soit x un nombre réel tel que pour tout nombre réel $\varepsilon > 0$, $x \leq \varepsilon$, alors $x \leq 0$. »

que l'on peut également écrire « $(\forall \varepsilon \in \mathbb{R}_+^*, x \leq \varepsilon) \Rightarrow (x \leq 0)$ ».

La démonstration par un raisonnement direct est difficile. Essayons de raisonner par contraposée. La contraposée de cette proposition est

$$(x > 0) \Rightarrow (\exists \varepsilon \in \mathbb{R}_+^*,\ x > \varepsilon),$$

ou

« Si x est strictement positif, alors il existe $\varepsilon > 0$ tel que $x > \varepsilon$. »

Cette propriété est facile à démontrer. En effet, si on prend un x strictement positif, on peut toujours trouver un ε. Si $x = 0.2$, alors $\varepsilon = 0.1$ convient. Si $x = 0.0000001$, alors $\varepsilon = 0.00000001$ convient. De manière général, $\varepsilon = x/2$ conviendra.

1.3.18 Remarque. Attention à ne pas confondre la contraposée $(\neg Q) \Rightarrow (\neg P)$ d'une implication avec sa réciproque $Q \Rightarrow P$. Cette dernière apparaît lorsque l'on veut démontrer la véracité de $P \Leftrightarrow Q$.

Par exemple, l'implication « $\forall x \in \mathbb{R},\ ((x > 2) \Rightarrow (x > 1))$ » est vraie. Sa contraposée est « $\forall x \in \mathbb{R},\ ((x \leq 2) \Rightarrow (x \leq 2))$ » est aussi vraie. En revanche, l'implication réciproque est « $\forall x \in \mathbb{R},\ ((x > 1) \Rightarrow (x > 2))$ » qui est fausse, car pour $x = 1,5$, on a bien $x > 1$ mais pas $x > 2$.

Preuve par raisonnement par l'absurde. Il consiste à démontrer une assertion en vérifiant que sa négation conduit à une contradiction avec les hypothèses. Formellement, si P désigne la proposition contenant les hypothèses, cela revient à démontrer $(P \wedge (\neq Q)) \Rightarrow (1 = 0)$ plutôt que de démontrer $P \Rightarrow Q$. Si il n'y a pas d'hypothèse et qu'on veut démontrer une assertion Q, cela revient plutôt à démontrer $\neg Q \Rightarrow (1 = 0)$, autrement dit la négation de Q amène à quelque chose de trivialement faux.

Par exemple, essayons de prouver que le nombre $\sqrt{2}$ est irrationnel, autrement dit $\sqrt{2} \notin \mathbb{Q}$. *Preuve :* Pour se faire, supposons que $\sqrt{2} \in \mathbb{Q}$, c'est-à-dire qu'il existe deux nombres entiers p et q avec $q \neq 0$ tels que

$$\sqrt{2} = \frac{p}{q}.$$

Quitte à simplifier la fraction, nous pouvons supposer que p et q n'ont pas de facteur commun, ou que $\text{pgcd}(p, q) = 1$, voir Chapitre 2. Si on multiplie par q et que l'on élève au carré des deux côtés, on obtient

$$2q^2 = p^2.$$

Le nombre à gauche est forcément pair, donc p^2 est pair, ce qui implique que p est pair (par contraposée, car si p était impair, p^2 serait impair...). Si p est pair, alors $p = 2k$ pour un certain k et $p^2 = 4k^2$ est un multiple de 4. On a alors

$$2q^2 = 4k^2 \Rightarrow q^2 = 2k^2$$

d'où q^2 est également pair, ce qui implique de même que q est pair. Au final, 2 est un facteur commun à p et à q, ce qui contredit les hypothèses que nous avions fixées. La supposition de départ est donc fausse, et $\sqrt{2} \notin \mathbb{Q}$.

Preuve par disjonction de cas. Dans certains raisonnements, on peut être amenés à considérer successivement plusieurs cas selon l'ensemble d'appartenance ou la valeur d'une certaine variable. Ainsi, si on veut démontrer une assertion « $(\forall x \in E, P(x))$ » pour un ensemble E qui est la réunion de deux ensembles A et B ($E = A \cup B$), on démontre d'abord l'assertion « $(\forall x \in A, P(x))$ », puis l'assertion « $(\forall x \in B, P(x))$ ». Ainsi, on peut alors conclure que l'assertion $P(x)$ est vraie pour tout $x \in E$, puisqu'on a traité tous les éléments de E dans A ou dans B.

Par exemple, démontrons que

« Pour tout entier naturel n, le nombre rationnel $\frac{n(n+1)}{2}$ est entier. »

Preuve : On va distinguer deux cas selon que n est pair ou impair. Rappelons que $\mathbb{N}$ est bien la réunion des entiers naturels pairs et des entiers naturels impairs.

Cas 1: n est pair. Si n est pair, alors il existe $k \in \mathbb{N}$ tel que $n = 2k$. D'où

$$\frac{n(n+1)}{2} = \frac{2k(n+1)}{2} = k(n+1) \in \mathbb{N}$$

puisque à la fois k et $(n+1)$ sont entiers.

Cas 2: n est impair. Alors on sait que l'entier $(n+1)$ est pair, d'où il existe $l \in \mathbb{N}$ tel que $n + 1 = 2l$ et donc

$$\frac{n(n+1)}{2} = \frac{n \times 2l}{2} = nl \in \mathbb{N}.$$

Au final, on a bien démontré l'assertion voulue pour tout entier.

Preuve par récurrence, ou induction. Ceci est un principe de preuve très couramment utilisé, notamment lorsque l'on manipule des propriétés à démontrer sur l'ensemble des entiers naturels.

Pour démontrer une assertion de la forme « $(\forall n \in \mathbb{N},\ P(n))$ », on va procéder en deux étapes:

- On démontre que $P(0)$ est vraie (ou $P(m)$ si la propriété est censée être vraie que pour tous les entiers à partir de m). Cette étape est appelée *l'initialisation*.
- On démontre ensuite« $(\forall n \in \mathbb{N},\ P(n))$ » que si $P(n)$ est vraie pour un certain n fixé, alors $P(n+1)$ est vraie, ce qui revient à démontrer la proposition

$$\forall n \in \mathbb{N}, (P(n) \Rightarrow P(n+1)).$$

  Cette étape est appelée *hérédité*.

1.3.19 Proposition. *Pour tout entier $n \geq 1$, on a*

$$P(n) : \quad \sum_{k=1}^{n} k = \frac{n(n+1)}{2}, \qquad \left(\textit{ou} \quad 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \right)$$

Proof. Commençons par l'initialisation : pour $n = 1$, la somme $\sum_{k=1}^{1}$ vaut 1, et $\frac{1(1+1)}{2} = 1$, donc la propriété $P(1)$ est vraie.

Démontrons à présent l'hérédité: supposons que la propriété $P(n)$ soit vraie pour un certain entier naturel n. On veut alors calculer la somme

$$\sum_{k=1}^{n+1} k = 1 + 2 + \cdots + n + (n+1) = (1 + 2 + \cdots + n) + (n+1).$$

En regroupant la somme de cette manière, on voit que la première partie correpond à $\sum_{k=1}^{n} k$, or par l'hypothèse $P(n)$ on sait que cette somme vaut $\frac{n(n+1)}{2}$. Donc, en remplaçant:

$$\begin{aligned}\sum_{k=1}^{n+1} k &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{n(n+1) + 2(n+1)}{2} \\ &= \frac{(n+2)(n+1)}{2} \\ &= \frac{(n+1)(n+2)}{2}\end{aligned}$$

où la deuxième égalité est une mise sur même dénominateur du deuxième terme, la troisième est une factorisation par le terme commun $(n+1)$ au numérateur, et la quatrième est une remise dans l'ordre des termes $(n+1)$ et $(n+2)$. Ceci démontre la propriété $P(n+1)$.

Ainsi, l'initialisation et l'hérédité étant prouvées, par le principe de récurrence on a démontré que la propriété est vraie pour tout entier naturel n. □

Chapitre 2

Arithmétique et applications

2.1. Nombres et bases

Les microprocesseurs sont capables d'effectuer des opérations arithmétiques de base (incluant addition, soustraction, et presque toujours multiplication et division euclidienne sur des petits entiers dans l'intervalle:

- $[-2^{t-1}, 2^{t-1}-1]$ pour des entiers signés;
- $[0, 2^t - 1]$ pour des entiers non signés

pour $t = 8, 16, 32, 64$ et parfois 128. En cas de dépassement de la taille mémoire, les calculs sont généralement effectués modulo 2^t: les logiciels de calcul doivent pouvoir travailler avec des entiers de taille plus grande, il faut les stocker en mémoire et implémenter les opérations arithmétiques de base en se ramenant à des petits entiers. Pour faire cela, on va écrire des entiers dans des *bases* adéquates.

2.1.1. Décomposition en base. Il s'agit de généraliser l'écriture des entiers qui nous est familière en base 10 à une base quelconque. En général, en informatique on utilisera 2 (binaire) ou une puissance de 2 (par exemple 16, l'hexadécimal). En fait, l'écriture du nombre 1602 signifique que

$$1602 = 1 \times 10^3 + 6 \times 10^2 + 0 \times 10^1 + 2 \times 10^0.$$

La taille d'un nombre est le nombre de chiffres qu'il contient, ainsi 1602 s'écrit avec 4 chiffres en base 10. Tous les nombres à 4 chiffres sont compris entre $1000 = 10^3$ et $9999 = 10^4 - 1$. Plus généralement, un nombre compris entre 10^{n-1} et $10^n - 1$ s'écrit avec n chiffres.

Cette décomposition en base 10 peut se généraliser pour un entier $b \geq 2$ quelconque:

2.1.2 Theorem. *On fixe un entier* $b \geq 2$. *Soit* N *un entier naturel non nul, on peut alors écrire* N *de manière unique sous la forme*

$$N = \sum_{i=0}^{n} N_i b^i$$

où les N_i *sont des coefficients dans* $[0, b-1]$ *et le dernier coefficient* N_n *est non nul. Ces coefficients* N_i *sont en général appelés* bits *si* $b = 2$, *ou* digits *si* $b = 16$. *On écrira alors*

$$N = (N_n\, N_{n-1}\, \ldots\, N_1\, N_0)_b$$

Dans ce théorème, l'entier $n+1$ est la taille du nombre N en base b, il est relié à N et b par la formule

$$n = \lfloor \frac{\log(N)}{\log b} \rfloor := \lfloor \log_b(N) \rfloor$$

où $\lfloor \cdot \rfloor$ désigne la fonction partie entière, log désigne la fonction logarithme en base 10 usuel, et $\log_b$ est appelé *logarithme en base* b. En effet, si N s'écrit avec n chiffres en base b, d'après ce qui précède on a

$$b^n \leq N < b^{n+1} \Leftrightarrow e^{n\log(b)} \leq N \leq e^{(n+1)\log(b)}$$

et en passant au logarithme:

$$n\log(b) \leq \log(N) < (n+1)\log(b) \Leftrightarrow n \leq \frac{\log(N)}{\log(b)} < n+1,$$

d'où le fait que n corresponde à la partie entière du terme au milieu.

2.1.3. De la base 10 vers une base b. Soit N un nombre écrit en base 10. Pour calculer la décomposition de N en base $b \geq 2$, on dispose d'un algorithme basé sur des divisions euclidiennes. Tous les coefficients (N_i) intervenant dans la décomposition de N en base b vont être calculés, en commençant par N_0, puis N_1, puis N_2, etc. et enfin N_n: ils sont donc calculés **de la droite vers la gauche** dans l'écriture ci-dessus.

- N_0 est le reste dans la division euclidienne de N par b;
- Ensuite, N reçoit le quotient de la division de N par b, et N_1 est le reste dans la division euclidienne de ce nouveau N par b.
- On continue ce procédé tant que N est non nul. Le dernier N obtenu supérieur à b sera le coefficient N_n.

2.1.4 Exemple.

- Conversissons 42 en base $b = 2$, en binaire. Voici les divisions euclidiennes successives:

  1. $42 = 21 \times 2 + 0$, donc $N_0 = 0$,
  2. $21 = 10 \times 2 + 1$, donc $N_1 = 1$,
  3. $10 = 5 \times 2 + 0$, donc $N_2 = 0$,
  4. $5 = 2 \times 2 + 1$, donc $N_3 = 1$,
  5. $2 = 1 \times 2 + 0$, donc $N_4 = 0$.
  6. $1 = 0 \times 2 + 1$, donc $N_5 = 1$.

  Finalement, on a $42 = (101010)_2$.

- Convertissons 1234 en base 16. Voici les divisions euclidiennes successives:

  1. $1234 = 77 \times 16 + 2$, donc $N_0 = 2$,
  2. $77 = 4 \times 16 + 16$, donc $N_1 = 13$,
  3. $4 = 0 \times 16 + 4$, donc $N_2 = 4$. Finalement, on a $1234 = (4\ 13\ 2)_{16}$.

2.1.5 Exercice. Écrire, en langage Python ou C, une fonction permettant de décomposer un nombre en base 10 dans une base b quelconque. Les deux paramètres seront entrés par l'utilisateur.

2.1.6 Solution.

```python
### Fonction qui convertit un nombre entier n en sa decomposition binaire b
def to_binary(n):
    b = ''
    while True:
        if n == 0:
            break
        elif (n % 2) == 0:
            n = n // 2
            b = '0' + b
        else:
            n = n // 2
            b = '1' + b
    return b

## Fonction qui prend une suite de bits, et renvoie l'entier associe en base 10
def decimal(b):
    b = b[::-1]
    d = 0
    power = 0
    for i in b:
        d += int(i) * (2 ** power)
        power +=1
    return d
```

2.1.7 Remarque. Notez que les décompositions en base 16, également appelés base *hexadécimale*, sont en général écrites d'une manière particulière. Les coefficients pouvant apparaître sont situés entre 0 et 15, et sont en général renommés selon le dictionnaire suivant:

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Ainsi, le nombre 1234 ci-dessus s'écrira plutôt en base 16 comme suit: $(4\ D\ 2)_{16}$.

2.1.8. D'une base b vers la base 10. Inversement, si on connaît l'écriture d'un nombre $N = (N_n\ N_{n-1}\ \dots\ N_1\ N_0)_b$ dans une base b, on l'obtient dans la base 10 en appliquant la formule

$$N = \sum_{i=0}^{n} N_i b^i.$$

Par exemple, pour $N = (4\ 13\ 2)_{16}$, on calcule N en base 10 par

$$N = 4 \times 16^2 + 13 \times 16 + 2 = 1234.$$

2.1.9. De la base 2 vers la base 16 et réciproquement. En général, si on souhaite convertir un nombre N écrit dans une base b dans une autre base b', on le convertira d'abord en base 10 puis on appliquera l'algorithme de divisions euclidiennes ci-dessus pour l'exprimer dans la base b'. Cependant, ce n'est pas toujours la méthode la plus rapide, notamment lorsque l'on veut passer d'une base b à une base b' de la forme b^k pour un certain entier $k \geq 2$.

Pour passer de la base $b = 2$ à la base $b' = 2^4$, il va suffire de regrouper les bits par paquets de 4, de la droite vers la gauche (quitte à compléter par des 0 à gauche pour avoir assez de bits dans le dernier paquet), et transcrire chacun de ces paquets de 4 bits par l'entier hexadécimal correspondant selon la traduction ci-dessus.

Par exemple, pour $N = (10101101010101100111110111)_2$, on obtient 7 paquets: de la droite vers la gauche (0111) correspondant à 7, (1111) correspondant à F, (1100) correspondant à C, (1010) correspondant à A, (1010) correspondant à A, (0101) correspondant à 5 et (0001) correspondant à 1. Donc

$$N = (10101101010101100111110111)_2 = (15AACF7)_{16}.$$

Inversement, si on souhaite passer d'une écriture en base 16 à une écriture en base 2, il suffit de considérer chaque *digit* de l'écriture en base 16, et de l'écrire en binaire **sur** 4 **bits**, quitte à rajouter des 0 à gauche si nécessaire. En fusionnant les paquets de 4 bits obtenus, on obtient l'écriture binaire. Reprenons par exemple $N = (4\ 13\ 2)_{16}$, on a 3 digits à convertir:

1. $(4)_{16} = (0100)_2$,
2. $(13)_{16} = (1101)_2$,
3. $(2)_{16} = (0010)_2$.

Finalement, on obtient

$$N = (4\ 13\ 2)_{16} = (010011010010)_2$$

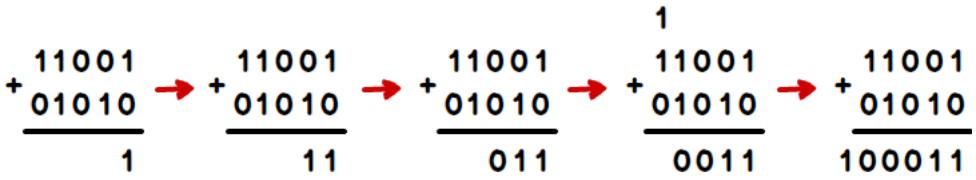
et on peut maintenant éventuellement supprimer les 0 inutiles tout à gauche.

2.1.10. Opérations arithmétiques. Si $N = \sum_{i=0}^{n} N_i b^i$ et $M = \sum_{i=0}^{n} M_i b^i$ sont deux entiers écrits en base b, alors

$$N + M = \sum_{i=0}^{\max(n,m)} (N_i + M_i) b^i$$

et pour obtenir la décomposition de $N + M$ en base b, il faut donc additionner les digits de l'écriture en base b. Cependant, on a a $N_i + M_i \in [0, 2b-2]$. Ainsi, si $N_i + M_i < b$, on peut garder ce digit sans problème mais si $N_i + M_i >$, le digit correspondant sera alors $N_i + M_i - b$, et on ajoutera 1 (une retenue, comme pour l'addition en base 10) à $N_{i+1} + M_{i+1}$, qui appartient à $[0, 2b-2+1]$, etc. Le nombre d'opérations à effectuer est de l'ordre du maximum du nombre de bits/chiffres/digits de N et M.

Voici par exemple le procédé d'addition de deux nombres binaires: les règles élémentaires sont les suivantes: $0+0=0$, $0+1=1=1+0$ et $1+1=0$ mais engendre une retenue sur le bit suivant. On peut ainsi poser une addition comme en base 10:



Si $N = \sum_{i=0}^{n} N_i b^i$ et $M = \sum_{i=0}^{n} M_i b^i$ sont deux entiers écrits en base b, alors

$$N \times M = \sum_{i=0}^{n} \sum_{j=0}^{m} N_i M_j b^{i+j}$$

Ici, si $b > 2$ par exemple $b = 10$, $N_i M_j$ sera très souvent supérieur à b, ce qui implique beaucoup de retenues, comme dans la multiplication classique en base 10. Le mieux est de regrouper les terms b^{i+j} ayant le même exposant, en utilisant des décalages pour tenir compte de b^i, comme lorsqu'on pose la multiplication en base 10 sur papier. On peut aussi additionner au fur et à mesure $N_i M_j$ au coefficient actuel de b_{i+j} tant qu'il est inférieur à b, et ajouter une retenue lorsqu'il devient supérieur. En binaire, si on pose la multiplication comme en base 10, il vaut mieux effectuer les additions une par une sinon la prise en compte des retenues est délicate.

2.1.11 Exemple. Voici comment poser la multiplication de $(11001)_2$ par $(11011)_2$:

$$\begin{array}{rcccccccc}
 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\times & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & . \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & . \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & . & . & . \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & . & . & . & . \\
\hline
 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}$$

2.1.12 Remarque. Le nombre de multiplications et d'additions dans le calcul de $N \times M$ est de l'ordre de nm. En ce qui concerne la taille du résultat: si on multiplie un nombre à 3 chiffres par un nombre à 2 chiffres, le résultat aura 4 ou 5 chiffres, penser par exemple à $412 \times 11 = 4532$ ou $412 \times 31 = 12772$. Dans le cas général,

$$b^n \leq N < b^{n+1}, \quad b^m \leq M \leq b^{m+1} \Rightarrow b^{n+m} \leq N \times M < b^{n+m+2}$$

et donc le résultat a pour taille $n + m + 1$ ou $n + m + 2$ chiffres. Si n et m sont proches, par exemple égaux, le résultat a pour taille $2n$ alors que l'algorithme de multiplication est en $O(n^2)$. On peut donc espérer l'existence d'algorithmes de multiplication plus rapides, par exemple l'algorithme de Karatsuba. Nous verrons également par la suite un algorithme rapide pour calculer des puissances de nombres.

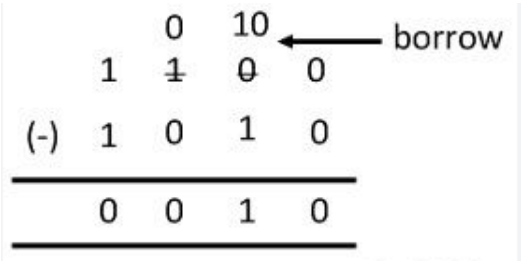
La soustraction de passe globalement comme l'addition, mais la gestion des retenues est différente lorsque le chiffre $N_i - M_i$ devient négatif. Nous allons illustrer ce procédé en binaire: les règles de soustraction sont les suivantes:

1. 1 - 0 donne 1,

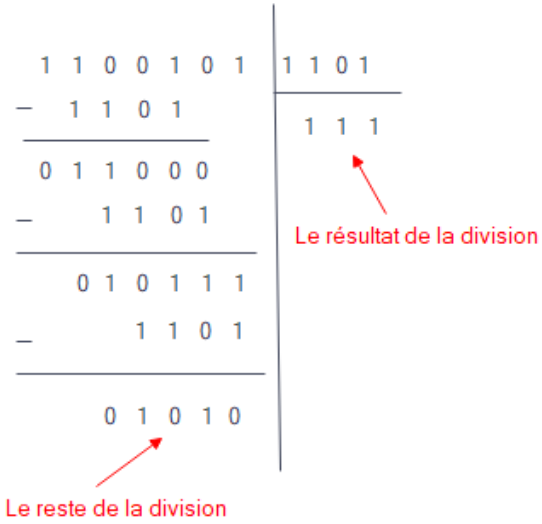
2. 1 - 1 donne 0,

3. 0 - 0 donne 0,

4. Si on a 0 - 1, il faudra faire une retenue sur le prochain bit précédent qui est non nul. On écrira alors le bit avec la retenue comme 10, qui correspond à 2 en binaire, avec la règle $10-1=1$, et ensuite le bit précédent qui était égal à 1 deviendra 0 avec la retenue. Notez qu'il faudra parfois faire plusieurs retenues successives.



Enfin, pour la division euclidienne, on dispose d'un algorithme dit *de la potence*, qui est celui que nous utilisons à l'école ! En binaire, il consiste à chercher les bits de l'écriture en base 2 du quotient en commençant par le bit de poids fort. La situation est simplifiée par rapport à un calcul en base 10 parce que le bit du quotient à déterminer ne peut être que 0 ou 1, il suffit de comparer avec le reste en cours.



2.1.13 Exercice. (1) Implémenter en Python l'algorithme d'addition et de soustraction binaire, pour des nombres a et b vus comme des listes de bits 0 ou 1.

(2) Implémenter l'algorithme de multiplication binaire.

(3) Implémenter l'algorithme de division euclidienne binaire.

2.1.14 Solution. (1) Addition et soustraction:

```
## Fonction qui renvoie l'addition de deux nombres a et b en base 2
def addition(a,b):
  max_len = max(len(a), len(b))
  a = a.zfill(max_len)
  b = b.zfill(max_len)
  #Cette commande de remplir a et b avec des zeros sur la gauche pour
   qu'ils aient le meme nombre de bits
  add = ''
  retenue = 0

  for i in range(max_len - 1, -1, -1):
        r = retenue
        r += 1 if a[i] == '1' else 0
        r += 1 if b[i] == '1' else 0
        add = ('1' if r % 2 == 1 else '0') + add

        retenue = 0 if r < 2 else 1

  if retenue != 0:
    add = '1' + add
  return add

#Fonction qui renvoie la soustraction de deux nombres a et b en base 2
def subtraction(a,b):
    max_len = max(len(a), len(b))

    a = a.zfill(max_len)
    b = b.zfill(max_len)

    sub = ''
    temp = 0

    for i in range(max_len - 1, - 1, - 1):
        num = int(a[i]) - int(b[i]) - temp
        if num % 2 == 1:
            sub = '1' + sub
        else:
            sub = '0' + sub

        if num < 0:
            temp = 1
        else:
            temp = 0

    if temp !=0:
        sub = '01' + sub

    if int(sub) == 0:
        sub = 0

    return sub
```

(2) Multiplication:

```
#Programme qui renvoie la multiplication de a par b en base 2
def multiplication(a, b):
    max_len = max(len(a), len(b))
    a = a.zfill(max_len)
```

```
    b = b.zfill(max_len)


    result = ''
    temp_result = ''

    temp = []
    zeroes = 0
    temp_index = 0

    for j in range(max_len - 1, - 1, - 1):

        temp_result = ''
        for i in range(max_len - 1, - 1, - 1):
            summ = int(a[i]) * int(b[j])
            if summ == 0:
                temp_result = '0' + temp_result
            elif summ == 1:
                temp_result = '1' + temp_result
        temp_result = temp_result + ('0' * zeroes)
        zeroes += 1

        temp.append(temp_result)

        if len(temp) == 2:
            result = addition(str(temp[0]), str(temp[1]))
        elif len(temp) > 2:
            temp_index = len(temp)
            temp_index += 1
            result = addition(str(result), str(temp[temp_index - 2]))
        else:
            pass
    return result
```

(3) Division euclidienne : quotient. Le reste est ensuite facile à calculer à partir des opérations précédentes.

```
#Fonction qui renvoie le quotient de a par b en base 2
def division(a,b):
  div=''
  temp='0'
  r=0

  for i in range(len(a)):
    if int(b) > int(temp):
      div += '0'
      temp += a[i]
    else:
      r=subtraction(temp,b)
      if int(r) == 0:
        temp = a[i]
        div += '1'
      else:
        r=str(r).lstrip('0')
        div += '1'
        temp = r + a[i]
  if int(temp) < int(b):
    div += '0'
```

```
    else:
      div += '1'

    return div.lstrip('0')
```

2.2. PGCD ET ALGORITHMES D'EUCLIDE

Dans cette Section, nous allons manipuler des propriétés importantes des nombres entiers relatifs, c'est-à-dire appartenant à $\mathbb{N}$ en général, et parfois $\mathbb{Z}$. Nous allons commencer par quelques définitions importantes sur les nombres premiers, et le PGCD.

2.2.1. Divisibilité. Soient a et b deux entiers appartenant à $\mathbb{Z}$. On dit que a *divise* b si l'une des propriétés équivalentes suivantes est vraie:

- la division de b par a donne comme résultat un nombre entier,
- le reste dans la division euclidienne de b par a donne 0.

On note alors $a \mid b$. On dit également que b est *divisible* par a, ou que b est un *multiple* de a. Si a ne divise pas b, on note $a \nmid b$. Par exemple, $2 \mid 6$, $3 \mid 18$ mais $3 \nmid 25$.

2.2.2. Nombres premiers. Un nombre entier naturel est dit *premier* si il n'est divisible que par 1 et lui-même (à signe près si on considère $\mathbb{Z}$ et non $\mathbb{N}$). On considère que 1 n'est par convention pas premier. 2 et 3 sont des nombres premiers, 4 ne l'est pas car 4 est divisible par 2.

Notez que pour trouver la liste des diviseurs d'un nombre n, il suffit de tester les divisions successives de n par i pour i allant de 1 à n, puisqu'un diviseur d'un nombre sera toujours inférieur à ce nombre. En fait, on peut même faire mieux, il suffit par symétrie de tester tous les i de 1 à $\lfloor \sqrt{n} \rfloor$. En effet, si $n = a \times b$ avec $a > \sqrt{n}$, alors nécessairement $b < \sqrt{n}$ et donc le test pour b donnera un diviseur, pour lequel il sera facile de trouver le second. On dispose donc d'un algorithme permettant de tester si un nombre n est premier: on va initialiser un booléen à vrai, puis, tester pour tous les i de 1 à $\lfloor \sqrt{n} \rfloor$ si i divise n. Si c'est le cas, on s'arrête et on renvoie faux. Si on arrive au bout de la boucle, on peut conclure que le nombre est premier. Par exemple, pour $n = 13$, on constate que $\sqrt{13} \sim 3,605$ et donc il suffit de tester les divisions de 13 par $1, 2$ et 3. Puisque ni 1, ni 2, ni 3 ne divise 13, on conclut que 13 est premier.

En général, on cherche plutôt à établir la liste des nombres premiers inférieurs à un certain $n \in \mathbb{N}$ qui sont premiers. Pour se faire, on dispose d'un algorithme appelé crible d'Ératosthène, qui consiste à partir de 2 et supprimer dans la liste des entiers de 2 à n tous les multiples de 2, puis recommencer avec l'entier suivant non supprimé, à savoir 3, et supprimer tous les multiples de 3, etc. Comme précédemment, on peut s'arrêter lorsqu'on arrive sur un entier m tel que $m > \sqrt{n}$, et alors les seuls nombres de la liste non rayés sont les nombres premiers.

2.2.3 Exercice. Implémenter en Python l'algorithme du crible d'Ératosthène pour lister les nombres premiers inférieurs à n, où n sera rentré par l'utilisateur.

2.2.4 Solution.

```
def erathostene(n):
  crible=[2]
  for i in range(3,n+1):
    prem=True
    for k in crible:
      if i%k==0:
        prem=False
    if prem == True:
      crible.append(i)

  return crible
```

2.2.5. PGCD. Soient a et b deux nombres entiers. Le *plus grand commun diviseur* (PGCD) de a et b est un nombre entier d tel que:

- d divise a et d divise b;
- d est le plus grand nombre qui divise à la fois a et b.

Le PGCD de a et b est noté $a \wedge b$. De manière immédiate, on a $a \wedge b = b \wedge a$. On dit que deux nombres a et b sont *premiers entre eux* si $a \wedge b = 1$. Nous allons présenter deux méthodes pour calculer le PGCD de deux nombres.

Méthode 1: Décomposition en facteurs premiers. Un résultat fondamental de l'arithmétique est que tout nombre entier peut se décomposer comme un produit de nombres premiers avec une certaine puissance:

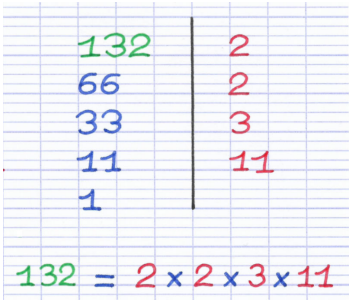
$$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$$

pour un certain nombre entier k, $p_1, \dots, p_k$ des nombres premiers et des entiers positifs $\alpha_1, \dots, \alpha_k$.

Pour écrire cette décomposition, il suffit comme ci-dessus de tester les divisions de n par les nombres premiers successifs, commençant par 2 puis:

- si n est divisible par 2, on note 2 dans la décomposition et on recommence avec $n/2$;
- si n n'est pas divisible par 2, on teste pour 3, puis 5, puis 7, etc.

Par exemple, pour $n = 42$, on a $42 = 2 \times 21$, puis on recommence avec 21. Il n'est pas divisible par 2 donc on teste avec 3, et on obtient $21 = 3 \times 7$. Donc $42 = 2 \times 3 \times 7$. La décomposition en facteurs premiers est en général représentée par un tableau à deux colonnes: la colonne de gauche représente les divisions successives de n par ses facteurs premiers, et celle de droite les facteurs premiers qui apparaissent.



Si on a établi la décomposition en facteurs premiers de nos deux nombres a et b, leur PGCD est obtenu de la façon suivante: on prend tous les facteurs premiers qui apparaissent **à la fois** dans la décomposition de a et de b, et on les considère à la puissance qui est la plus petite entre celle de a et de b.

2.2.6 Exemple. (1) PGCD de 124 et 66:

$$124 = 2 \times 62 = 2 \times 2 \times 31 = 2^2 \times 31, \qquad 66 = 2 \times 33 = 2 \times 3 \times 11$$

donc $124 \wedge 66 = 2$.

(2) PGCD de 280 et 8:

$$280 = 2 \times 140 = 2^2 \times 70 = 2^3 \times 35 = 2^3 \times 5 \times 7, \qquad 8 = 2^3$$

donc $280 \wedge 8 = 8$.

Méthode 2: Algorithme d'Euclide. L'algorithme d'Euclide est un algorithme fondamental de l'arithmétique. Il permet de calculer le PGCD de deux nombres en se basant sur le fait suivant: si a et b sont deux nombres entiers (où on suppose $a > b$), alors

$$a \wedge b = (a - b) \wedge b$$

et plus généralement, si $a = b \times q + r$ est la division euclidienne de a par b, alors

$$a \wedge b = b \wedge r.$$

Ainsi, pour déterminer le PGCD de a et b, on va réaliser des divisions euclidiennes successives, en commençant par la division euclidienne de a par b: $a = b \times q + r$, puis a reçoit b, b reçoit r et on recommence. **Le PGCD de a et b est alors le dernier reste non nul apparaissant dans cette suite de divisions euclidiennes.** Formellement, on va construire une suite (r_n) d'entiers strictement décroissante définir par $r_0 = a$, $r_1 = b$ et

pour $n \geq 1$, r_{n+1} est le reste dans la division euclidienne de r_{n-1} par r_n (d'où $r_{n+1} < r_n$).

Puisque cette suite est décroissante, il existe un rang k tel que $r_{k+1} = 0$, on a alors $a \wedge b = r_k$.

2.2.7 Exemple. (1) PGCD de 21 et 15. Divisions euclidiennes successives:

1. $21 = 15 \times 1 + 6$,
2. $15 = 6 \times 2 + 3$,
3. $6 = 3 \times 2 + 0$, et on s'arrête car on a un reste nul. Le PGCD est le dernier reste non nul, donc 3 (cf ligne précédente), et $21 \wedge 15 = 3$.

(2) PGCD de 216 et 22. Divisions euclidiennes successives:

1. $216 = 22 \times 9 + 18$,
2. $22 = 18 \times 1 + 4$,
3. $18 = 4 \times 4 + 2$,
4. $4 = 2 \times 2 + 0$, et on s'arrête car on a un reste nul. Le PGCD est le dernier reste non nul, donc 2 (cf ligne précédente), et $216 \wedge 22 = 2$.

2.2.8 Exercice. (1) Implémenter en Python une fonction qui permet de donner la décomposition en facteurs premiers d'un nombre entier.

(2) Implémenter en Python l'algorithme d'Euclide avec une fonction récursive. On rappellera que le quotient de la division euclidienne de a par b est obtenu par `a // b` et le reste de la division euclidienne de a par b par `a % b`.

(3) Implémenter une version itérative de l'algorithme d'Euclide.

2.2.9 Solution. (1) Décomposition en facteurs premiers :

```
def decomp_prem(n):
  res=[]
  d=2
  while n%d==0:
    res.append(d)
    q=int (n/d)
    n=q
  d=3
  while d<=n:
    while n%d==0:
      res.append(d)
      q=int (n/d)
      n=q
    d=d+2
  return res
```

(2) Algorithme d'Euclide récursif:

```
def euclide_rec(a,b):
    #Renvoie le PGCD des entiers a et b, version recursive
    if b==0:
        return a
    else:
        r=a%b
        return pgcd(b,r)
```

(3) Algorithme d'Euclide itératif :

```
def pgcd_ite(a,b):
  #Renvoie le PGCD des entiers a et b, version iterative
  while (b !=0):
    r=a%b
    a=b
    b=r
  return a
```

On peut aussi mentionner le résultat important suivant, qui est une conséquence direct de la décomposition en facteurs premiers ou de l'algorithme d'Euclide.

2.2.10 Proposition. *Soient a et b deux entiers naturels non nuls, si a divise b alors les diviseurs communs de a et de b sont les diviseurs de a, et*

$$a \wedge b = a.$$

Et enfin, un dernier résultat souvent utile concernant le PGCD:

2.2.11 Proposition. *Soient a et b deux entiers naturels non nuls. Si $d = a \wedge b$, alors il existe deux entiers naturels a' et b' premiers entre eux tels que $a = d \times a'$ et $b = d \times b'$.*

2.2.12. PPCM. Soient a et b deux nombres entiers. Le *plus petit commun multiple* (PPCM) de a et b est un nombre entier m tel que:

- a divise m et b divise m;
- m est le plus petit nombre qui est un multiple à la fois de a et de b.

Le PPCM de a et b est noté $a \vee b$. On a de même $a \vee b = b \vee a$. L'algorithme de décomposition en facteurs premiers donne une méthode de calcul du PPCM de a et b: on prend les facteurs premiers qui apparaissent dans la décomposition de a et dans la décomposition de b (pas forcément les deux !) et si ils sont dans les deux, on prend la plus grande puissance dans a ou b.

2.2.13 Exemple. PPCM de 124 et 66: on a déjà vu que

$$124 = 2^2 \times 31, \qquad 66 = 2 \times 3 \times 11,$$

d'où $124 \vee 66 = 2^2 \times 3 \times 11 \times 31 = 4092$.

On dispose aussi de la propriété importante suivante :

2.2.14 Proposition. *Soient* a *et* b *deux entiers naturels non nuls, alors*

$$a \times b = (a \wedge b) \times (a \vee b).$$