BTS SIO

Mathématiques pour l'Informatique

Lycée Rabelais
Saint Brieuc
2023-2024

Partie I

Mathématiques

Chapitre 1

Bases de numération

« Partons sur de bonnes bases. »

On note $\mathbf{N}$ l'ensemble des *entiers naturels* : $\mathbf{N} = \{0; 1; 2; ...\}$.

Nous avons l'habitude d'utiliser la base 10 pour représenter les entiers naturels, c'est-à-dire qu'on utilise 10 symboles, appelés *chiffres* pour les écrire : 0, 1, 2, ..., 9. Or il n'en a pas toujours été ainsi :

- au I^{er} millénaire av. J.-C., les Babyloniens utilisaient la base soixante pour mesurer le temps et les angles ;
- durant le I^{er} millénaire, les Mayas et les Aztèques se servaient de la base vingt (et d'ailleurs en France, 80 se lit « quatre-vingts ») ;
- entre le VII^{e} et le XV^{e} siècle, les astronomes arabes utilisaient la base cent cinquante pour élaborer des tables permettant de trouver la position d'un astre dans le ciel à un moment donné.

De nos jours, en Informatique, on utilise beaucoup la base deux, dite *binaire* et la base seize, appelée *hexadécimale*.
L'objectif de ce chapitre est de donner les méthodes permettant d'écrire un entier naturel dans une base donnée, plus précisément dans les bases 2, 10 et 16. Nous verrons également comment passer facilement du binaire à l'hexadécimal et vice-versa.

1 Écriture binaire d'un entier naturel

1.1 Pourquoi le binaire ?

Pour simplifier, disons qu'au niveau le plus « bas » d'un ordinateur, se trouvent des (millions de) transistors qui jouent chacun un rôle d'interrupteur. De multiples points de l'ordinateur peuvent alors être soumis à une tension (état 1) ou non (état 0). En considérant 2 de ces points, on voit que l'état de ce système peut être 00, 01, 10 ou 11. Cela fait 4 possibilités et le binaire est né !

1.2 Comprendre l'écriture en base 2

Puisqu'il n'y a que deux chiffres en binaire, compter est simple mais nécessite rapidement plus de chiffres qu'en base 10 :

Écriture décimale	0	1	2	3	4	5	6	7	8	...
Écriture binaire	0	1	10	11	100	101	110	111	1000	...

Notation

On écrira $(11)_{10}$ pour insister sur le fait qu'on parle du nombre 11 *en base 10*, et $(11)_2$ pour dire que c'est une écriture binaire.
Lorsque ce n'est pas précisé cela veut dire que l'écriture est en base 10.
Ainsi $(11)_2 = 3$, et de même, $(111)_2 = 7$.

Tout entier naturel admet une unique écriture décimale (c'est-à-dire en base 10), il en va de même en binaire :

Propriété : écriture binaire d'un entier naturel

Tout entier naturel possède une unique écriture en base 2, dite *écriture binaire*. Plus précisément, soit $n \in \mathbf{N}$, alors il existe un unique entier $k \in \mathbf{N}$ et $k+1$ nombres a_i, uniques et valant 0 ou 1 et tels que

$$n = a_0 2^0 + a_1 2^1 + \ldots + a_k 2^k$$

ce qui s'écrit aussi

$$n = \sum_{i=0}^{k} a_i 2^i$$

Exemple

Lorsqu'on regarde le tableau précédent, on voit que $6 = (110)_2$.
Cela s'interprète ainsi :

Chiffre binaire	1	1	0
Valeur	2^2	2^1	2^0

et on obtient $6 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2$.

Méthode 1 : passer de la base 2 à la base 10

Que vaut $(11101)_2$?

Chiffre binaire	1	1	1	0	1
Valeur	2^4	2^3	2^2	2^1	2^0

$$\begin{aligned}(11101)_2 &= 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 16 + 8 + 4 + 1 \\ &= 29\end{aligned}$$

Exercice 1

1. Donner l'écriture décimale des onze premières puissances de deux.
2. Donner l'écriture décimale de $(1101)_2$ et $(1\,1010)_2$

Méthode 2 : passer de la base 10 à la base 2

Comprenons ce que veut dire une écriture décimale :

$$\begin{aligned}203 &= 200 + 3\\ &= 2 \times 10^2 + 0 \times 10^1 + 3 \times 10^0\end{aligned}$$

Faisons la même chose en base 2 :

$$\begin{aligned}203 &= 128 + 64 + 8 + 2 + 1\\ &= 2^7 + 2^6 + 2^3 + 2^1 + 2^0\\ &= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0\\ &= (11001011)_2\end{aligned}$$

Cette méthode est pratique quand l'entier est petit et que l'on connaît bien les premières puissances de deux.
Quand ce n'est pas le cas, une autre méthode (la méthode 3) peut être employée.

Évidemment, PYTHON connait le binaire et travaille avec des valeurs entières de type `int` (*integer* veut dire « entier » en Anglais, pour plus de précisions, voir le chapitre 10, partie 2).
Voici comment écrire un `int` en binaire et comment obtenir l'écriture binaire d'un `int`.

Python

```
>>> 0b11001011 # faire précéder le nombre de 0b
203
>>> bin(29)
'0b11101'
```

Exercice 2

Appliquer la méthode 2 pour donner les écritures binaires des nombres suivants.

a. 6 ;

b. 26 ;

c. 126 ;

d. 1026.

1.3 Un algorithme pour déterminer l'écriture binaire d'un entier naturel

Méthode 3 : les divisions successives

Voici comment on trouve les chiffres de l'écriture *décimale* de 203 :

On divise 203 par 10, cela fait 20, il reste 3, c'est le chiffre des unités.
On recommence avec 20, on le divise par 10, cela fait 2, reste 0, chiffre des dizaines.
On continue, on divise 2 par 10, cela fait 0, reste 2, chiffre des des centaines.
Puisqu'on a trouvé un quotient de 0, on s'arrête.

On peut écrire cela simplement :

$$\begin{array}{rl|l} 203 & 10 & \\ \cline{2-3} 3 & 20 & 10 \\ \cline{2-3} 0 & 2 & 10 \\ \cline{2-3} 2 & 0 & \end{array}$$

Voici maintenant comment on trouve son écriture binaire. On procède comme en base 10 mais en divisant par 2 :

$$\begin{array}{rl|l} 203 & 2 & \\ \cline{2-3} 1 & 101 & 2 \\ \cline{2-3} 1 & 50 & 2 \\ \cline{2-3} 0 & 25 & 2 \\ \cline{2-3} 1 & 12 & 2 \\ \cline{2-3} 0 & 6 & 2 \\ \cline{2-3} 0 & 3 & 2 \\ \cline{2-3} 1 & 1 & 2 \\ \cline{2-3} 1 & 0 & \end{array}$$

On a donc successivement établi :

$$\begin{aligned} 203 &= 101 \times 2 + 1 \\ &= (50 \times 2 + 1) \times 2 + 1 \\ &= ((25 \times 2 + 0) \times 2 + 1) \times 2 + 1 \\ &= (((12 \times 2 + 1) \times 2 + 0) \times 2 + 1) \times 2 + 1 \\ &= ((((6 \times 2 + 0) \times 2 + 1) \times 2 + 0) \times 2 + 1) \times 2 + 1 \\ &= (((((3 \times 2 + 0) \times 2 + 0) \times 2 + 1) \times 2 + 0) \times 2 + 1) \times 2 + 1 \\ &= ((((((1 \times 2 + 1) \times 2 + 0) \times 2 + 0) \times 2 + 1) \times 2 + 0) \times 2 + 1) \times 2 + 1 \\ &= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= (11001011)_2 \end{aligned}$$

Cette succession d'égalités n'est (heureusement) pas à écrire à chaque fois.

Exercice 3

Utiliser la méthode 3 pour retrouver les écritures binaires de l'exercice précédent.

Les trois méthodes précédentes se programment facilement et la dernière est de loin la plus courte à écrire.

1.4 Vocabulaire

Un chiffre décimal peut être 0, 1, 2, 3, 4, 5, 6, 7, 8 ou 9.
Un chiffre binaire peut être seulement 0 ou 1. En Anglais, *chiffre binaire* se traduit par *binary digit*, que l'on abrège en *bit*. On garde cette dénomination en Français.

Le bit est « le plus petit morceau d'information numérique » .

Pour les écrire, on regroupe les chiffres décimaux par paquets de 3, comme dans 1 230 014 par exemple. En binaire on groupe les bits par 4, on écrira donc $17 = (1\ 0001)_2$.

La plupart du temps, en machine, les bits sont groupés par 8 (deux paquets de 4). Un tel paquet s'appelle

un *octet*, et on écrit donc des *mots binaires* de longueur 8 tels que 0000 0011 : l'octet représente ici le nombre 3. Les bits à zéros ne sont pas inutiles.

Lorsqu'on considère un nombre écrit en binaire, on parle souvent de *bit de poids fort* et de *bit de poids faible* pour parler respectivement du bit associé à la plus grande puissance de 2, et du bit d'unités. Considérons l'octet $(0010\ 0101)_2$. Son bit de poids fort est 0, son bit de poids faible est 1.

2 Écriture hexadécimale d'un entier naturel

La base « naturelle » de l'informatique est la base 2, mais elle n'est pas très pratique car elle donne lieu à des écritures trop longues. La base 10 nous paraît bien meilleure parce que nous avons l'habitude de l'utiliser, mais elle ne fait pas bon ménage avec la base 2 : il n'y a pas de méthode simple pour passer du décimal au binaire, et vice versa.
La base 16, ou base *hexadécimale*, est en revanche très adaptée à l'écriture des paquets de 4 bits, et par extension à celle des octets et autres écritures binaires.

En hexadécimal, on dispose de 16 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.

Propriété : écriture binaire d'un entier naturel

Tout entier naturel possède une unique écriture en base 16, dite *écriture hexadécimale*. Plus précisément, soit $n \in \mathbf{N}$, alors il existe un unique entier $k \in \mathbf{N}$ et $k+1$ nombres a_i, uniques et valant 0, 1, 2, ..., ou F et tels que

$$n = a_0 16^0 + a_1 6^1 + \ldots + a_k 16^k$$

ce qui s'écrit aussi

$$n = \sum_{i=0}^{k} a_i 16^i$$

Remarque

On a vu une propriété similaire en base 2 et en fait elle est valable *dans toutes les bases b* (où b est un entier naturel supérieur ou égal à 2). Cela justifie par exemple l'utilisation de la base 20 ou de la base 150.

Les méthodes que l'on a vu en base 2 et 10 se transposent en base 16.

Méthode 4 : passer de la base 16 à la base 10

Déterminons l'écriture décimale de $(D4A)_{16}$:

$$\begin{aligned}(D4A)_{16} &= 13 \times 16^2 + 4 \times 6 + 10 \times 16^0 \text{ car D vaut 13 et A vaut 10.}\\ &= 3402\end{aligned}$$

Méthode 5 : passer de la base 10 à la base 16

Déterminons maintenant l'écriture hexadécimale de 503 :

503	16		
7	31	16	
15	1	16	
	1	0	

$503 = 31 \times 16 + \underline{7}$.

$31 = 1 \times 16 + \underline{15}$ et 15 s'écrit $\underline{F}$.

$1 = 0 \times 16 + \underline{1}$ et on arrête car le quotient est nul.

$503 = (1F7)_{16}$.

Un éditeur hexadécimal montre le contenu d'un fichier, d'un disque dur, ou de la RAM d'un ordinateur. La première colonne indique l'adresse, puis 16 octets écrits en hexadécimal et enfin les caractères correspondants.

3 Hexadécimal et binaire : un mariage heureux

Le grand avantage qu'apporte l'hexadécimal s'illustre facilement :

Méthode 6 : passer de la base 2 à la base 16

$$\begin{aligned}(101101000011101)_2 &= (0101\ 1010\ 0001\ 1101)_2 \\ &= \left(\underbrace{0101}_{5}\ \underbrace{1010}_{A}\ \underbrace{0001}_{1}\ \underbrace{1101}_{D}\right)_2 \\ &= (5A1D)_{16}\end{aligned}$$

Méthode 7 : passer de la base 16 à la base 2

$$(F7B)_{16} = \left(\underbrace{1111}_{F}\ \underbrace{0111}_{7}\ \underbrace{1011}_{B}\right)_2$$

4 Additions

On pose l'opération à la main : c'est la même chose qu'en base 10.

x	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E	20
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D	30
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C	40
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B	50
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A	60
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69	70
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78	80
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87	90
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96	A0
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5	B0
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4	C0
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3	D0
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2	E0
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1	F0
10	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	100

Table de multiplication hexadécimale.

En base 2

La seule différence avec la base 10 c'est que deux 1 donnent $(2)_{10}$ donc $(10)_2$, donc un zéro et une retenue de 1. Quand il y a deux 1 et une retenue de 1 en plus, cela donne $(3)_{10}$ donc $(11)_2$, donc un 1 et une retenue de 1.

Exemple

				1	1	1		
	1	1	0	1	0	1	0	1
+				1	1	1	0	0
=	1	1	1	1	0	0	0	1

En base 16

C'est encore la même chose. Il faut bien se souvenir de la valeur de A, B, C, D, E et F.
Ajouter 8 et 3 ne provoque pas de retenue puisque $8 + 3 = 11$ et que 11 est B en base 16.
Dès que l'addition de 2 chiffres dépasse 15, il y aura une retenue : par exemple 9 et A donnent $(19)_{10}$, donc $(13)_{16}$. Ainsi on note 3 et une retenue de 1.

Exemple

		1			
	2	4	9	8
+	1	7	A	3
=	3	C	3	B

5 Multiplications par 2 en binaire

Propriété

- Multiplier un nombre écrit en binaire par 2 revient à décaler la virgule d'un cran vers la droite (ou ajouter un zéro à droite si le nombre est entier).
- Diviser un nombre écrit en binaire par 2 revient à décaler la virgule d'un cran vers la gauche.

Exemples

- $(1\,0101)_2 \times (2)_{10} = (10\,1010)_2$
- $(11,01)_2 \times (16)_{10} = (11\,0100)_2$
- $(1\,1101)_2 \div (2)_{10} = (1110,1)_2$
- $(101)_2 \div (32)_{10} = (0,0010\,1)_2$

Remarque

Rappelons-nous que $(2)_{10} = (10)_2$, et que plus généralement 2^n s'écrit en binaire comme « un 1 suivi de n zéros » .

6 Exercices

Exercice 4

1. Calculer $2^6 + 2^4 + 2^3 + 2^0$.
2. En déduire l'écriture binaire de 89.

Exercice 5

1. Calculer $2^7 + 2^3 + 2^2 + 2^1$.
2. En déduire l'écriture décimale de $(10001110)_2$.

Exercice 6

En utilisant la méthode 2, donner l'écriture binaire de

1. 56
2. 35
3. 13

Exercice 7

En utilisant la méthode 3, donner l'écriture binaire de

1. 142
2. 273
3. 1000

Exercice 8

- Donner l'écriture décimale de $(1101\,1010)_2$.
- Donner l'écriture binaire de 2016.
- Donner l'écriture hexadécimale de 2016.

Exercice 9

- Donner les écritures décimales de $(11)_2$, $(111)_2$, $(1111)_2$.
- Soit $n \in \mathbf{N}$, conjecturer la valeur de $\left(\underbrace{1\ldots1}_{n \text{ chiffres}}\right)_2$.

Exercice 10

Pour multiplier par dix un entier naturel exprimé en base dix, il suffit d'ajouter un 0 à sa droite, par exemple, $12 \times 10 = 120$.
Quelle est l'opération équivalente pour les entiers naturels exprimés en base deux ?

Exercice 11

1. Donner l'écriture binaire de 174.
2. Donner celle de 17.
3. Poser l'addition de 174 et 17 en binaire.
4. Donner l'écriture décimale du résultat et vérifier.

Exercice 12

1. Donner l'écriture hexadécimale de 1022.
2. Donner celle de 3489.
3. Poser l'addition de 1022 et 3489 en hexadécimal.
4. Donner l'écriture décimale du résultat et vérifier.

Exercice 13

Ajouter $(1101\,1011)_2$ et $(0011\,0110)_2$ en posant l'opération.

Exercice 14

On pose $a = (1001)_2$, $b = (0010\,1000)_2$ et $c = (0001\,0111)_2$.

Calculer $a + b + c$ en posant les opérations (on peut faire des étapes ou bien tout calculer en une fois).

Exercice 15

On reprend les données de l'exercice précédent. Calculer $a \times 4 + b \div 8 + c$.

Exercice 16

Calculer à la main $(AF3)_{16} + (8AD)_{16}$.

Exercice 17

Calculer à la main $(123)_{16} + (456)_{16} + (789)_{16}$ en posant une seule opérations si possible.

Exercice 18*

Le Roi d'un pays imaginaire fait frapper sa monnaie par 8 nains : chacun d'entre eux produit des pièces d'or de 10g chacune.

Un jour, son Mage lui annonce : « Majesté, mon miroir magique m'a prévenu que certains de vos nains vous volent. Ils prélèvent 1g d'or sur chaque pièce qu'ils frappent. Pour vous aider à trouver les voleurs, voici une balance magique. Elle est précise au gramme près et peut peser autant que vous voulez. Malheureusement elle ne peut être utilisée qu'une fois. »

Le lendemain, le Roi convoque les 8 nains en demandant à chacun d'apporter un coffre rempli de pièces d'or qu'il a frappées.
On suppose que

- chaque nain dispose d'autant de pièces que nécessaire ;
- un nain honnête n'a que des pièces de 10g ;
- un nain voleur n'a que des pièces de 9g ;

Peux-tu aider le Roi pour démasquer les voleurs ?

Exercice 19

On dispose d'une clé de cryptage notée $(a_1, a_2, a_3, a_4, a_5) = (63, 62, 65, 68, 34)$.
Cette clé, publiée par la personne destinataire, permet à quiconque de lui envoyer un message crypté. Voici comment on crypte le message.
On associe d'abord à chaque lettre son rang dans l'alphabet, selon la correspondance suivante :

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
Rang	1	2	3	4	5	6	7	8	9	10	11	12	13
Lettre	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Rang	14	15	16	17	18	19	20	21	22	23	24	25	26

Pour crypter une lettre :

- on détermine son rang à l'aide du tableau de correspondance précédent ;
- on écrit ce nombre en base 2 sur 5 bits ; on ainsi obtient 5 chiffres $(m_1, m_2, m_3, m_4, m_5)$, chaque chiffre étant égal à 0 ou à 1 ;
- on détermine alors la valeur cryptée, égale à la somme $\sigma = a_1m_1 + a_2m_2 + a_3m_3 + a_4m_4 + a_5m_5$.

On remarque qu'une lettre est ainsi cryptée par un nombre entier.
Exemple : on veut crypter la lettre « I ».

- Le rang de I est 9 ;
- on écrit ce nombre en base deux sur 5 bits : $9_{10} = 8 + 1 = (01001)_2$,

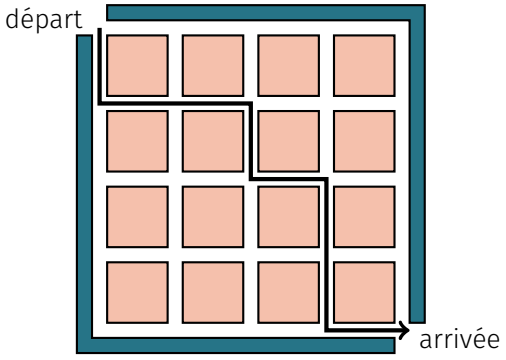
– on calcule la somme $\sigma = 0 \times 63 + 1 \times 62 + 0 \times 65 + 0 \times 68 + 1 \times 34 = 96$.

La lettre « I » est donc cryptée par l'entier 96.

Question : crypter la lettre « W » selon cette méthode.

Exercice 20 : un petit jeu

On veut programmer un petit jeu : une bille commence dans la grille suivante, toujours à la case départ.
On doit l'amener à la case arrivée en appuyant seulement sur les touches $\rightarrow$ et $\downarrow$ du clavier. Voici un exemple de partie :



Pour représenter les différents parcours, on associe à chacun d'entre eux un entier de la manière suivante :

– on note la séquence de touches pressées dans l'ordre ;

– on remplace chaque $\downarrow$ par 0 et chaque $\rightarrow$ par 1 ;

– on obtient l'écriture binaire de l'entier final.

1. Justifier que l'entier obtenu à partir de l'exemple vaut 105.
2. Combien faut-il de bits pour représenter un chemin donné ?
3. Représenter le parcours associé au nombre 85.
4. On considère un entier N représentant un chemin.
   a. Est-il possible d'avoir $N = 31$?
   b. Rappeler ce que vaut $(\underbrace{1\ldots1}_{\text{k bits}})_2$.
   c. Quel est le plus petit entier N qui représente un chemin ? Dessiner le chemin associé.
   d. Quel est le plus grand ? Dessiner le chemin associé.

Exercice 21 : additions en base 2

Les additions « à la main » en base 2 s'effectuent de la même manière qu'en base 10, la seule différence c'est que deux 1 donnent $(2)_{10}$ donc $(10)_2$, donc un zéro et une retenue de 1. Quand il y a deux 1 et une retenue de 1 en plus, cela donne $(3)_{10}$ donc $(11)_2$, donc un 1 et une retenue de 1. Par exemple, on veut calculer $(1101\ 0101)_2 + (1\ 1100)_2$, on pose l'opération en mettant les retenues en rouge.

				1	1			
	1	1	0	1	0	1	0	1
+				1	1	1	0	0
=	1	1	1	1	0	0	0	1

Le résultat est donc $(1111\ 0001)_2$.

1. Donner les écritures décimales de $(1101\ 0101)_2$, $(1\ 1100)_2$, $(1111\ 0001)_2$ et vérifier qu'il n'y a pas d'erreur d'addition.
2. Vérifier que l'on a bien l'égalité $138 + 111 = 249$ lorsque l'on effectue l'addition en binaire.
3. Calculer $(1011\ 1101)_2 + (11)_2$ et donner les écritures décimales des 3 nombres intervenant dans cette addition.

Exercice 22 : entiers non signés sur un octet

Dans certains langages, on utilise un octet (c'est-à-dire 8 bits) pour stocker un entier positif. On stocke dans l'octet l'écriture binaire de l'entier, telle quelle.

– en C et C++, ce type de variable s'appelle `unsigned char` ;

– en C#, il s'appelle `byte`.

1. Quels sont les entiers représentables de cette manière ?
2. Pour ajouter deux valeurs de ce type, on effectue l'addition en binaire, mais s'il y a une retenue à reporter à la fin (qui irait théoriquement dans un 9ème bit) on ne la prend pas en compte : on ne garde que les 8 premiers bits du résultats en partant de la droite.
   a. Ajouter avec cette méthode 129 et 130 et donner le résultat sous forme décimale.
   b. Ci dessus on a écrit un petit programme en C++. Dire ce qu'il fait et interpréter le résultat affiché dans la console.

```
#include <iostream> // bibliothèque d'affichage
int main() // début de la fonction principale
{
    unsigned char c = 0; // on définit la variable c
    for (int i = 0; i < 300; i++) // on fait une boucle pour
    {
        std::cout << "valeur de i : " << i; // on affiche la valeur
↪   de i
        std::cout << "  et valeur de c : " << (int)c; // on affiche
↪   la valeur de c en base 10
        std::cout << endl; // on revient à la ligne (END Line)
        c++; // on augmente c de 1
    }
    return 0; // la fonction principale renvoie zéro par convention
}
```

Voilà ce que la console affiche :

```
valeur de i : 0 et valeur de c : 0
valeur de i : 1 et valeur de c : 1
et cætera
valeur de i : 254 et valeur de c : 254
valeur de i : 255 et valeur de c : 255
valeur de i : 256 et valeur de c : 0
valeur de i : 257 et valeur de c : 1
et cætera
valeur de i : 298 et valeur de c : 42
valeur de i : 299 et valeur de c : 43
```

Exercice 23 : entiers signés en complément à 2 sur 8 bits

Pour représenter un entier signé (positif ou négatif) sur un octet, on peut penser au système suivant : le bit de poids fort vaut 0 si l'entier est positif et 1 s'il est négatif. Les 7 autres bits servent à représenter la partie numérique du nombre.
Par exemple,

$$(1001\ 1010)_2$$

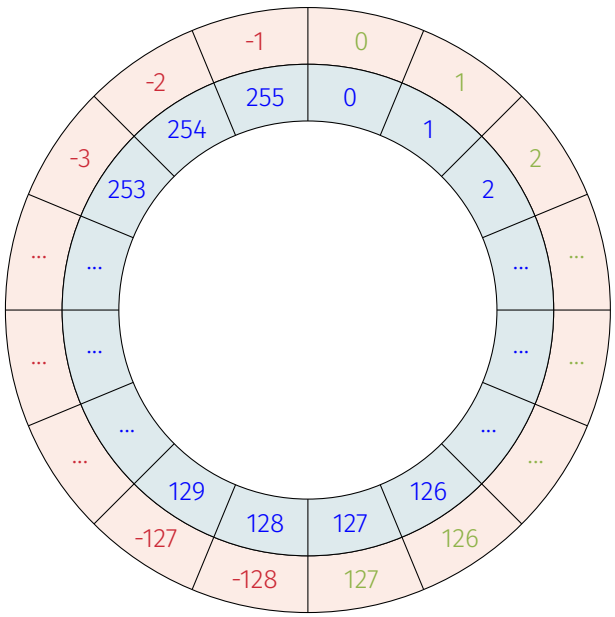
représente un nombre négatif car son bit de poids fort est à 1. Sa partie numérique est $(1\ 1010)_2 = 26$, donc il représente -26.

1. **a.** Que représente $(0000\ 0000)_2$? Et $(1000\ 0000)_2$?

   b. Donner la représentation de +26 et ajouter les représentations de +26 et -26 en binaire. Quel nombre représente le résultat obtenu ? Est-ce cohérent ?

2. Pour pallier ces problèmes, on représente les entiers *en complément à 2 sur 8 bits* :

   - si on veut représenter un entier compris entre 0 et $2^7 - 1 = 127$, alors on le représente tel quel sur un octet ;
   - si on veut représenter un entier n compris entre $-2^8 = -128$ et -1, alors on le représente par $n + 256$.



Ce schéma représente l'encodage des entiers relatifs en complément à 2 sur 8 bits (que l'on retrouve dans le type `char` en C et C++).

En bleu figurent les nombres tels qu'ils sont stockés dans la mémoire, sur un octet.
En orange figurent les nombres représentés par les nombres bleus :

- lorsqu'il est compris entre 0 et 127, un nombre bleu représente ce même nombre (affiché en vert) ;
- lorsqu'il est compris entre 128 et 255, un nombre bleu p représente le nombre $p - 256$ (affiché en rouge).

a. Donner la représentation en complément à 2 sur 8 bits du nombre +7.

b. Faire de même avec -3.

c. Ajouter en binaire ces deux représentations (une éventuelle retenue après le 8^{e} bit est ignorée), quel nombre cela représente-t-il ?

3. Recommencer la question précédente et vérifier que l'égalité $-12 + 8 = -4$ est vérifiée en complément à 2 sur 8 bits.

4. Le programme suivant affiche -116. Expliquer ce résultat.

```
#include <iostream> // nécessaire pour utiliser cout
int main() // début de la fonction main
{
    char c1 = 120; // on définit une première variable
    char c2 = 20; // puis une deuxième
    char c3 = c1+c2; // on les ajoute
    std::cout << (int) c3; // on affiche le résultat en base 10
    return 0; // la fonction main renvoie traditionnellement zéro
}
```

Exercice 24 : masque jetable

Le but de cet exercice est d'étudier une méthode de cryptage inventée par Gilbert Vernam en 1917, et appelée « masque jetable ». Dans tout l'exercice, on note respectivement M le mot initial, K la clé de cryptage et Y le mot crypté. Les trois nombres M, K, Y sont des entiers naturels. Les chiffres hexadécimaux sont notés 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

1. Questions préliminaires :

   a. Donner la représentation en hexadécimal de l'entier binaire 1011101_2.

   b. Calculer en travaillant dans le système hexadécimal les sommes $(7)_{16} + (4)_{16}$ et $(A)_{16} + (C)_{16}$.

2. Soit M et K deux entiers naturels écrits en hexadécimal, tels que la longueur de l'écriture de K est supérieure ou égale à celle de M, et tels que l'écriture de K ne comporte aucun chiffre 0.

   Pour crypter le mot M avec la clé K, on procède comme suit : pour chaque chiffre m du mot initial M, on considère le chiffre k de la clé K qui a la même position que m dans l'écriture.

   On obtient alors le chiffre y du mot crypté Y qui a la même position que m dans l'écriture du mot initial M, de la façon suivante : y est le chiffre hexadécimal des unités de la somme $m + k$.

Le mot crypté Y est déterminé en hexadécimal par la juxtaposition dans le même ordre des chiffres y calculés pour chaque chiffre m du mot M.

Exemple : avec $M = (49)_{16}$ et $K = (19)_{16}$

- Avec le chiffre de rang 1 en partant de la droite : $m = 9$ et $k = 9$; donc $m + k = (12)_{16}$ et par suite $y = 2$;
- avec le chiffre de rang 2 : $m = 4$ et $k = 1$; donc $m + k = (5)_{16}$ et par suite $y = 5$.

Donc le mot crypté est $Y = (52)_{16}$.

Question : avec le mot initial $M = (7A)_{16}$ et la clé $K = (4C)_{16}$, déterminer le mot crypté Y.

3. Par cette méthode, on admet que le décryptage suit les mêmes étapes en remplaçant la clé K par une autre clé K′. Lorsque l'écriture de K comporte au maximum deux chiffres hexadécimaux, la clé K′ est l'écriture en hexadécimal de la différence (écrite en décimal) $(272)_{10} - (K)_{10}$.

   Cette question est une question à choix multiple. Une seule réponse est exacte. Recopier sur la copie seulement la réponse exacte. On ne demande pas de justification.

   Avec la clé de cryptage $K = (19)_{16}$, la clé de décryptage K′ est égale à :

   Réponse A : $(253)_{16}$ **Réponse B** : $(247)_{16}$
   Réponse C : $(FD)_{16}$ **Réponse D** : $(F7)_{16}$

Chapitre 2

Écriture des « réels »

« Tout cela est-il bien réel ? »

1 Écriture décimale et arrondi

On sait que tout nombre réel admet une écriture décimale :

- « un et demi » s'écrit 1,500 000... et on enlève les zéros inutiles, cela fait 1, 5.
- « trois septièmes » s'écrit 0,428 571 428 571 428 571 428 571 4 ... et pour insister sur le fait que le motif 428571 se répète indéfiniment on écrit $0,\underline{428\,571}$.
- π a une écriture décimale qui commence par 3,141 592 653 59 mais son écriture décimale comporte une infinité de chiffres *sans qu'aucun motif ne se répète*.

On est souvent amenés à *arrondir* les nombres réels : soit le nombre n'est « pas très grand » et on ne veut garder que quelques chiffres après la virgule, soit il est « plutôt grand » et on ne veut garder que quelques chiffres significatifs :

Méthode 1

On veut arrondir $\frac{3}{7}$ à 10^{-3} près, c'est-à-dire à 3 chiffres après la virgule.
Il y a trois possibilités :

- **Arrondi par défaut** : On « coupe » après le troisième chiffre :

$$\frac{3}{7} \approx 0,428 \quad \text{à } 10^{-3} \text{ près par défaut.}$$

- **Arrondi par excès** : On « coupe » après le troisième chiffre et on ajoute 10^{-3}, c'est-à-dire un *millième* :

$$\frac{3}{7} \approx 0,429 \quad \text{à } 10^{-3} \text{ près par excès.}$$

- **Arrondi au plus près** : On regarde le chiffre immédiatement après le troisième (celui qui correspond à 10^{-4}). Si c'est 0,1,2,3 ou 4, on prend l'arrondi par défaut, si c'est 5,6,7,8 ou 9, on prend l'arrondi par excès.

$$\frac{3}{7} \approx 0,429 \quad \text{à } 10^{-3} \text{ au plus proche.}$$

Exercice 25

Utiliser la méthode 1 pour déterminer

1. L'arrondi de $\frac{13}{11}$ à 10^{-2} près par défaut

2. L'arrondi de $\sqrt{2}$ à 10^{-4} près par excès.

3. L'arrondi de $\frac{149}{999}$ à 10^{-3} au plus près.

Méthode 2

On veut arrondir 273 692,291 à 10^4, c'est à dire à la dizaine de milliers.
Là encore il y a trois possibilités. On commence par remarquer que le chiffre correspondant à 10^4 est le 7.

- **Arrondi par défaut** : On remplace tous les chiffres à droite du 7 par des zéros. La partie décimale disparaît.
$$273\,692{,}291 \approx 270\,000 \quad \text{à } 10^4 \text{ près par défaut.}$$
- **Arrondi par défaut** : On prend l'arrondi par défaut et on ajoute 10^4.
$$273\,692{,}291 \approx 280\,000 \quad \text{à } 10^4 \text{ près par excès.}$$
- **Arrondi au plus près** : On regarde le chiffre immédiatement après celui de 10^4 (celui qui correspond à 10^3). Si c'est 0,1,2,3 ou 4, on prend l'arrondi par défaut, si c'est 5,6,7,8 ou 9, on prend l'arrondi par excès.
$$273\,692{,}291 \approx 270\,000 \quad \text{à } 10^4 \text{ au plus proche.}$$

Exercice 26

Utiliser la méthode 2 pour déterminer

1. L'arrondi de 38 564 526 à 10^3 près par défaut

2. L'arrondi de 281 564 526 à 10^8 près par excès.

3. L'arrondi de 9 524 à 10^3 au plus près.

2 Écriture dyadique et arrondi

2.1 Écriture dyadique

Lorsqu'on écrit un nombre décimal tel que 3,719, on a l'égalité suivante :

$$3{,}719 = 3 \times 10^0 + 7 \times 10^{-1} + 1 \times 10^{-2} + 9 \times 10^{-3}$$

Il est possible de faire la même chose en base 2 : on ajoute des puissances de 2 d'exposants négatifs.

Méthode : retrouver l'écriture décimale à partir d'une écriture dyadique

On considère le nombre $n = (1010,011)_2$. Quelle est son écriture décimale ?

- Sa partie entière est $(1010)_2$, ce qui vaut 10.
- Sa partie décimale est

$$\begin{aligned}(0,011)_2 &= 2^{-2} + 2^{-3} \\ &= 0,25 + 0,125 \\ &= 0,375\end{aligned}$$

Ainsi

$$n = 10,375$$

Exercice 27

Utiliser la méthode précédente pour déterminer les écriture décimales de

1. $(0,101)_2$
2. $(11,01)_2$
3. $(1111,1111)_2$

Méthode : construire un nombre dyadique

On veut l'écriture du nombre $5,75$ en base 2. Pour la partie entière, c'est simple :

$$5 = (101)_2$$

Pour la partie décimale, on remarque que

$$\begin{aligned} 0,75 &= 0,5 + 0,25 \\ &= \tfrac{1}{2} + \tfrac{1}{4} \\ &= 2^{-1} + 2^{-2} \end{aligned}$$

Ainsi

$$0,75 = (0,11)_2$$

Finalement

$$5,75 = (101,11)_2$$

Exercice 28

Utiliser la méthode précédente pour déterminer les écriture dyadiques de

1. $3,25$
2. $12,625$
3. $7,8125$

Remarque

Un nombre décimal n'a pas généralement une écriture dyadique « qui se termine » .
Par exemple 0,1 (qui est pourtant le nombre décimal le plus simple auquel on puisse penser) s'écrit

$$0,1 = (0,\ 0001\ 1001\ 1001\ \underline{1001}\ldots)_2$$

2.2 Arrondi

Pour arrondir un nombre en base 2, on fait pareil qu'en base 10 :

Exemples

1. Arrondissons $n = (1101\,1010)_2$ à 2^4 au plus proche :
   L'arrondi par défaut est $(1101\,0000)_2$ mais comme le bit juste à droite du bit de 2^4 est un 1, il faut rajouter 2^4 à notre valeur arrondie, pour avoir la valeur par excès qui, elle, est plus proche :

$$\begin{array}{cccccccccc} & & & & 1 & & & & & \\ & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline = & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}$$

   Ainsi $n \approx (1110\,0000)_2$ à 2^4 au plus proche.

2. Pour les nombres dyadiques, c'est la même chose. Arrondissons $m = (11,\ 0011\,1001)_2$ à 2^{-5} au plus proche :
   Le bit de 2^{-6} est un 0, donc la bonne valeur arrondie est par défaut :

   $$m \approx (11,\ 0011\,1)_2 \text{ à } 2^{-5} \text{ au plus proche.}$$

Remarque

Quand on nous demande d'arrondir sans préciser, on convient que c'est au plus proche.

Exercice 29

Utiliser la méthode précédente pour déterminer

1. L'arrondi de $(1101\,1010)_2$ à 2^3 près par défaut.
2. L'arrondi de $(1, 1110\,101)_2$ à 2^{-3} près par excès.
3. L'arrondi de $(0, 001)$ à 2^{-1} au plus proche.

Exercice 30

Donner l'écriture décimale des nombres suivants

1. $(101, 1)_2$
2. $(1, 011)_2$
3. $(0, 1111\,111)_2$ en remarquant que c'est « $(111\,1111)_2$ divisé par 2^7.

Exercice 31

Écrire en base 2 les nombres suivants :

1. 3,5
2. 7,75
3. 27,625

Exercice 32

1. Quel est l'arrondi de $(10,011)_2$ à $(0,1)_2$ près ?
2. Quel est l'arrondi de $(11011)_2$ à $(100)_2$ près ?
3. Quel est l'arrondi de $(0,001011)_2$ à $(0,0001)_2$ près ?

Exercice 33

1. Quel est l'arrondi de $(11,0101)_2$ à 2^{-2} près ?
2. Quel est l'arrondi de $(10111011)_2$ à 32 près près ?

Exercice 34**

On aimerait trouver l'écriture dyadique (illimitée) de $\frac{1}{3}$. On note donc

$$\frac{1}{3} = (0, a_1 a_2 a_3 \ldots)_2$$

où a_i vaut 1 ou 0.

1. Expliquer pourquoi a_1 vaut *nécessairement 0*.
2. On note $x = \frac{1}{3}$. Montrer que x vérifie $4x = 1 + x$.
3. Quelle est l'écriture dyadique de $4x$?
4. Quelle est celle de $1 + x$?
5. En écrivant que ces 2 écritures représentent le même nombre, en déduire que

$$\frac{1}{3} = (0,0101\ 0101\ldots)_2$$

Chapitre 3

Arithmétique modulaire

1 Entiers naturels et division euclidienne

Définition (rappel) : ensemble des entiers naturels

On note $\mathbf{N}$ l'ensemble des *entiers naturels*.

$$\mathbf{N} = \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12 \ldots\}$$

Définition (rappel) : division euclidienne dans $\mathbb{N}$

Soient A et B deux entiers naturels, et $B \neq 0$. Il existe deux nombres uniques Q et R (vérifiant $0 \leqslant R < B$) tels que l'on puisse écrire

$$A = Q \times B + R$$

C'est exactement la division que l'on a apprise en sixième (celle où l'on s'arrête aux nombres entiers) :

A	B
R	Q

- A est appelé le *dividende* ;
- B est le *diviseur* ;
- Q est le *quotient* ;
- R est le *reste*, il est *impérativement* plus petit que B.

Exemples

Effectuons la division euclidienne de 27 par 7 :

27	7
6	3

Ainsi on a $\underset{A}{27} = \underset{Q}{3} \times \underset{B}{7} + \underset{R}{6}$.

Effectuons la division euclidienne de 297 par 11 :

297	11
0	27

Ainsi on a $\underbrace{286}_{A} = \underbrace{26}_{Q} \times \underbrace{11}_{B} + \underbrace{0}_{R}$.

Exercice 35

Effectuer les divisions euclidiennes suivantes :

a. 28 par 8 **b.** 100 par 9 **c.** 379 par 11 **d.** 427 par 13

Définition : diviseurs et multiples

Quand la division euclidienne de A par B donne un reste nul (lorsqu'elle tombe juste) on dit que *B est un diviseur de A* ou bien (ce qui revient au même) que *A est un multiple de B*.

Exemples

- $27 = 3 \times 7 + 4$ donc 7 ne divise pas 27 (7 n'est pas *un diviseur* de 27).
  On peut dire aussi que 27 n'est pas multiple de 7.
- $286 = 26 \times 11 + 0$ donc 1 est donc un diviseur de 286.

Remarques

a. $28 = 4 \times 5 + 8$ mais ce n'est pas la division euclidienne de 28 par 5 car le « reste » 8 n'est pas plus petit que 5.

$$\begin{aligned} 28 &= 4 \times 5 + 8 \\ &= 4 \times 5 + 5 + 3 \\ &= 5 \times 5 + 3 \end{aligned}$$ voilà la division euclidienne.

b. *« une division euclidienne n'en donne pas toujours 2 »* :
$27 = 3 \times 7 + 4$ est la division euclidienne de 27 par 7.
On peut réarranger : $27 = 7 \times 3 + 4$ mais ce n'est pas la division euclidienne de 27 par 3 car le reste est trop grand.

Ceci dit quand le reste est suffisamment petit, on obtient « 2 divisions pour le prix d'une » :
$162 = 12 \times 13 + 1$ est la division euclidienne de 162 par 12, et de 162 par 13 aussi.

c. *« une division euclidienne qui tombe juste en donne deux en général »* :
$297 = 27 \times 11$ nous donne 2 diviseurs de 297 : 27 et 11.
$16 = 4 \times 4$ aussi... mais c'est deux fois le même !

d. avec la calculatrice, pour savoir si B divise A il suffit d'entrer $A \div B$ et de regarder si le résultat est entier.

Exercice 36

Les égalités suivantes, peuvent-elles être des divisions euclidiennes ?
Si oui, préciser A, B, Q et R.

a. $65 = 32 \times 2 + 1$

b. $80 = 5 \times 10 + 30$

c. $100 = 9 \times 11 + 1$

d. $17 = 3 \times 4 + 5$

Exercice 37

Les égalités suivantes ne sont pas des divisions euclidiennes.
Transformez-les pour qu'elles le deviennent (il peut y avoir plusieurs possibilités).

a. $19 = 3 \times 4 + 7$

b. $30 = 2 \times 10 + 10$

c. $29 = 4 \times 5 + 9$

d. $23 = 4 \times 7 - 5$

Python

```
>>> a = 17   # déclare une variable a de type int (entier)
>>> b = 5    # idem avec b
>>> q = a // b # // donne le quotient par b

>>> q
5

>>> r = a % b  # %  donne le reste modulo b
>>> r
2
```

2 Diviseurs et nombres premiers

Propriété

Soient a et b deux entiers naturels. Dire que b est un diviseur de a veut dire que la division euclidienne de a par b donne un reste nul.
Cela signifie donc qu'il existe un entier naturel k tel que

$$a = k \times b$$

On peut également dire que a est multiple de b.

Remarque

Si b divise a il existe un entier naturel k tel que

$$a = k \times b$$

et donc k divise a également.

Exemple

$20 = 5 \times 4$ donc 5 et 4 sont deux diviseurs de 20.

Exercice 38

À l'aide de la calculatrice (ou non), déterminer si b divise a.

a. $a = 251$ et $b = 13$

b. $a = 8$ et $b = 80$

c. $a = 111$ et $b = 37$

d. $a = 131\,072$ et $b = 8\,192$

Propriétés

- Si a est divisible par b alors tout multiple de a est également divisible par b.
- Si a est divisible par b et que b est divisible par c alors a est divisible par c.
- Si a et b sont divisibles par c alors a+b aussi et (si a>b) a-b aussi.

Exemples

a. 12 est divisible par 3 donc tout multiple de 12 aussi : 12 000 est donc divisible par 3.

b. 120 est divisible par 12 et 12 est divisible par 3 donc 120 est divisible par 3.

c. Puisque 12 000 et 12 sont divisibles par 3, 12 000-12, soit 11 988, l'est aussi.

Définition : entier naturel premier

Un entier naturel est dit *premier* lorsqu'il admet 2 diviseurs *distincts* : 1 et lui-même.

- 0 n'est pas premier : $0 = 1 \times 0 = 2 \times 0 = 3 \times 0 = \ldots$
- 1 n'a qu'un diviseur : lui-même. Il n'est pas premier.
- 2 est premier.
- 3 aussi.
- 4 ne l'est pas car 1, 2 et 4 divisent 1.

Il est assez simple de montrer qu'il y a une infinité de nombres premiers. Les nombres premiers jouent un rôle très important en mathématiques et interviennent dans les systèmes de cryptographie (basiques ou sophistiqués).

Exercice 39 : crible d'Ératosthène

Dans la grille suivante :

- Entourer 2 et barrer 2 et tous ses multiples.
- Une fois cela fait, 3 est le premier nombre non barré après 2 donc on l'entoure et on barre tous ses multiples.
- Continuer jusqu'à ce que tous les nombres de la grilles soient traités (soit barrés soit entourés).
- Les nombres entourés sont *des nombres premiers*.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Recopier ici la liste des nombres premiers trouvés :

Propriété

Soit n un entier supérieur ou égal à 2 :

- ou bien n possède un diviseur inférieur ou égal à $\sqrt{n}$ et, à ce moment là, *n n'est pas premier.*
- ou bien *aucun nombre premier* inférieur à $\sqrt{n}$ ne divise n et *il est premier.*

Méthode : déterminer si un nombre est premier ou non

Voici le début de la liste des nombres premiers :

$$\{2;\ 3;\ 5;\ 7;\ 11;\ 13;\ 17;\ 19\}$$

- 133 est-il premier ?

  $\sqrt{133} \approx 11,5$ donc on regarde si 133 est divisible par 2, 3, 5, 7 ou 11.
  On trouve que $133 = 19 \times 7$ donc 133 n'est pas premier.

- 251 est-il- premier ?

  $\sqrt{251} \approx 15,8$ donc on regarde si 251 est divisible par 2, 3, 5, 7, 11, ou 13.
  Ce n'est pas le cas : 251 est donc premier.

Exercice 40

Les nombres suivants sont ils premiers ?

a. 143

b. 145

c. 141

d. 147

e. 247

f. 257

Propriété : décomposition en produit de facteurs premiers

Tout entier naturel supérieur ou égal à 2 se décompose de manière unique (à l'ordre près) en *produit de facteurs premiers*.

Méthode

Pour décomposer un nombre en produit de facteurs premiers, on cherche n'abord ses petits diviseurs premiers et on recommence jusqu'à trouver 1 :

234 |	2	on voit que 234 est pair
117 |	3	car 3 est le plus petit nombre premier qui divise 117
39 |	3	et ainsi de suite
13 |	13	...
1 |		on arrête

On a donc

$234 = 2 \times 3 \times 3 \times 13$
$= 2 \times 3^2 \times 13$ et c'est la décomposition en produit de facteurs premiers de 234.

Exercice 41

Décomposer les nombres suivants en produit de facteurs premiers.

a. 30

b. 60

c. 96

d. 684

Méthode : liste des diviseurs d'un entier

Pour trouver *tous* les divieurs d'un entier supérieur ou égal à 2 :

a. on le décompose en produit de facteurs premiers.

b. on écrit tous les nombres que l'on peut former en prenant « moins de facteurs » dans cette décomposition.

Exemple

Trouvons tous les diviseurs de 60 :
$60 = 2 \times 30 = 2^2 \times 15 = 2^2 \times 3 \times 5$. Ses diviseurs sont donc

1	$2^0 \times 3^0 \times 5^0$
5	$2^0 \times 3^0 \times 5^1$
3	$2^0 \times 3^1 \times 5^0$
15	$2^0 \times 3^1 \times 5^1$
2	$2^1 \times 3^0 \times 5^0$
10	$2^1 \times 3^0 \times 5^1$
6	$2^1 \times 3^1 \times 5^0$
30	$2^1 \times 3^1 \times 5^1$
4	$2^2 \times 3^0 \times 5^0$
20	$2^2 \times 3^0 \times 5^1$
12	$2^2 \times 3^1 \times 5^0$
60	$2^2 \times 3^1 \times 5^1$

Il est très facile d'oublier des diviseurs. Pour que cela n'arrive pas il faut utiliser une méthode logique pour les énumérer.
Voici un exemple d'algorithme qui les donne tous, en reprenant l'exemple de 60.

Algorithme

```
Variables
i, j, k : entiers
Début
    Pour i allant de 0 à 2
        Pour j allant de 0 à 1
            Pour k allant de 0 à 1
                Afficher 2 ^ i * 3 ^ j * 5 ^ k
            FinPour
        FinPour
    FinPour
Fin
```

Exercice 42

Donner la liste des diviseurs des nombres suivants.

a. 30

b. 25

c. 96

d. 684

3 pgcd de deux entiers naturels non nuls

Deux entiers naturels non nuls ont au moins un diviseur commun : 1. Parmi tous les nombres qui les divisent tous les deux il y en a un plus grand : leur pgcd.

Définition

Soient a et b deux entiers naturels non nuls.
On note $pgcd(a;b)$ et on lit « pgcd de a et de b » le plus grand entier qui divise à la fois a et b.

Exemples

Le plus grand nombre qui divise à la fois 12 et 16, c'est 4. Ainsi $pgcd(12;16) = 4$.
25 et 27 n'ont aucun diviseur commun plus grand que 1 : $pgcd(25;27) = 1$.

Définition

Lorsque $pgcd(a;b) = 1$ on dit que a et b sont *premiers entre eux*.

Exemples

25 et 27 sont premiers entre eux. 8 et 15 aussi.

Remarque

Il ne faut pas confondre *nombre premier* (tout court) et *nombres premiers entre eux* :

- 25 et 27 sont premiers entre eux mais aucun de ces deux nombres n'est premier.
- 3 et 30 ne sont pas premiers entre eux : leur pgcd vaut 3. Pourtant 3 est premier.

Méthode

Soient a et b deux entiers naturels que l'on a décomposés en produit de facteur premiers :

- S'ils n'ont aucun facteur commun alors ils sont premiers entres eux.
- Sinon, on fait le produit des facteurs communs avec la plus petite puissance qui apparaît dans chacune des décompositions.

Exemple

On veut $pgcd(240;72)$.

- On commence par décomposer 240 : $240 = 2^4 \times 3^1 \times 5^1$.
- On fait de même pour 72 : $72 = 2^3 \times 3^2$
- Il y a deux facteurs premiers en commun dans cette décomposition : 2 et 3. On garde à chaque fois l'exposant le plus petit et on en fait le produit :

$$pgcd(a;b) = 2^3 \times 3^1 = 24$$

Exercice 43

En utilisant les décompositions en produit de facteurs premiers, donner le PGCD de

a. 15 et 27

b. 63 et 99

c. 222 et 148

d. 192 et 69

Propriété

Soient a et b 2 entiers non nuls. Si $b < a$ et qu'on effectue la division euclidienne de a par b, on obtient

$$a = q \times b + r$$

et à ce moment là

$$pgcd(a;b) = pgcd(b;r)$$

Méthode : Algorithme d'Euclide

Cette méthode se base sur la propriété précédente. On veut trouver le pgcd de 420 et 182.

$420 = 2 \times 182 + 56$ donc $pgcd(420;182) = pgcd(182,56)$ et on recommence.
$182 = = 3 \times 56 + 14$ donc $pgcd(182,56) = pgcd(56,14)$ et on poursuit.
$56 = 3 \times 14 + \boxed{0}$ et on s'arrête.

La dernière ligne nous indique que $pgcd(56;14) = 14$, ainsi $pgcd(420;182) = 14$.

Exercice 44

En utilisant la méthode de votre choix, donner le PGCD de

a. 198 et 256

b. 546 et 230

c. 180 et 105

d. 357 et 399

Exercice 45

En utilisant l'algorithme d'Euclide, donner le PGCD de

a. 130 et 85

b. 4114 et 1530

c. 882 et 540

d. 1725 et 1309

Exercice 46

On dispose de 280 roses rouges et 490 roses blanches, avec lesquelles on veut faire le plus grand nombre possible de bouquets identiques.
Combien peut-on faire de tels bouquets et quelle est la composition de chacun d'eux ?

Exercice 47

Une feuille A4 a pour dimensions 21 cm et 29,7 cm. Alice cherche à savoir comment elle peut quadriller sa feuille à l'aide de carrés de mêmes dimensions, qui soient les plus gros possibles.
Quelle sera la taille des carrés ? Combien en fera-t-elle ?

4 Congruences

Définition

Soit n un entier naturel non nul et a et b deux entiers naturels.
On dit que a et b sont *congrus modulo n* si les divisions euclidiennes de a et b par n donnent le même reste.
On écrit cela

$$a \equiv b \quad [n]$$

Exemple

a. Prenons deux multiples de 5, ils sont tous congrus modulo 5 puisque lorsqu'on les divise par 5 le reste est nul.

$$15 \equiv 20 \quad [5]$$

b. Ajoutons leur 2 à tous les deux, ils sont encore congrus modulo 5 puisque lorsqu'on les divise par 5 le reste est 2.

$$17 \equiv 22 \quad [5]$$

c. Dans la vie courante, on raisonne parfois *modulo* 12 :

$16 = 1 \times 12 + 4$
$16 \equiv 4 \quad [12]$

Et de même $17 \equiv 5 \quad [12]$ et $18 \equiv 6 \quad [12]$: « 5 heures de l'après-midi, c'est 17 :00 » et cætera.

Propriété

Soient a et b deux entiers naturels tels que $a > b$.
Dire que $a \equiv b \quad [n]$ revient à dire que $a - b$ est un multiple de n.

Exemples

a. On a vu que $17 \equiv 22 \quad [5]$, et en effet $22 - 17 = 5$.

b. Partons de 11 et ajoutons lui un multiple de 3 : $11 + 7 \times 3 = 32$.
11 et 32 sont congrus modulo 3 : la différence est $32 - 11 = 7 \times 3$, et en faisant les divisions euclidiennes on trouve : $11 = 3 \times 3 + 2$ et $32 = 10 \times 3 + 2$ donc 11 et 32 on le même reste dans la division euclidienne par 3.

Exercice 48

Ci dessous, il y a 4 congruences. Dire si elles sont vraies ou non :

a. En faisant les « divisions euclidiennes par le modulo » .

b. En regardant si la différence des deux nombres est un « multiple du modulo » .

Quelle est-la méthode la plus rapide ?

a. $19 \equiv 13 \quad [6]$

b. $53 \equiv 29 \quad [5]$

c. $28 \equiv 0 \quad [7]$

d. $257 \equiv 353 \quad [32]$

Propriété : compatibilité avec les opérations

Soient a, b, c, d, n et p 5 entiers naturels, et n non nul.

Supposons que $a \equiv b \quad [n]$ et $c \equiv d \quad [n]$. Alors

$$a + p \equiv b + p \quad [n]$$
$$a + c \equiv b + d \quad [n]$$
$$a - c \equiv b - d \quad [n]$$

$$a \times p \equiv b \times p \quad [n]$$
$$a \times c \equiv b \times d \quad [n]$$
$$a^p \equiv b^p \quad [n]$$

Exemples

a. Par quel nombre se termine $123456789 \times 981234567$?
$123456789 = 12345678 \times 10 + 9$ donc le premier nombre est congru à 9 modulo 10.
De même le deuxième est congru à 7 modulo 10.
Donc leur produit est congru à $9 \times 7 = 63$ modulo 10, donc 3 modulo 10.
Ainsi $123456789 \times 981234567$ se termine par 3.

b. Que vaut 1314 modulo 13 ?

$$\begin{aligned} 1314 &= 13 \times 100 + 14 \\ &\equiv 0 \times 100 + 1 \quad [13] \\ &\equiv 1 \quad [13] \end{aligned}$$

Exercice 49

1. Vérifier que $90 \equiv 6 \quad [7]$ et que $66 \equiv 3 \quad [7]$.
2. En utilisant les propriétés des congruences, compléter les résultats suivants en mettant l'entier naturel le plus petit possible :

a. $90 + 66 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [7]$
b. $90 \times 66 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [7]$
c. $902 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [7]$
d. $663 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [7]$

Exercice 50

1. Faire les divisions euclidiennes de 200 et de 900 par 13 et traduire les résultats en congruences.
2. En utilisant les propriétés des congruences, compléter les résultats suivants en mettant l'entier naturel le plus petit possible :

a. $200 + 900 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [13]$
b. $200 \times 900 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [13]$
c. $2002 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [13]$
d. $9003 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [13]$
e. $2900 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [13]$
f. $9413 \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [13]$

Propriété

Modulo n, les multiples de a sont les multiples de $pgcd(a, n)$.

Méthode

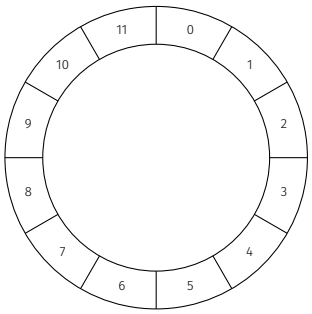
Soit une liste `L` de longueur 90, dont les éléments sont `L[0]`, `L[1]` ...`L[89]`.
On la parcourt en commençant par `L[0]` et en ajoutant 50 à chaque fois, modulo 90, indéfiniment.
Alors, puisque $pgcd(50, 90) = 10$, les multiples de 50 modulo 90 sont les multiples de 10 modulo 90 : cela veut dire qu'on ne parcourra pas tous les éléments de la liste, mais seulement :
`L[0]`, `L[10]`, `L[20]`, `L[30]`, `L[40]`, `L[50]`, `L[60]`, `L[70]` `L[80]`, `L[90]`.

Remarque

Si on parcourt une liste de longueur n en faisant des « sauts de p indices modulo n » alors on ne parcourra l'ensemble de la liste que si n et p sont premiers entre eux.

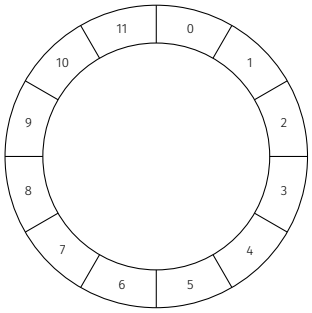
Exercice 51 : parcours d'une liste circulaire à pas constant

On considère le motif suivant : les cases sont numérotés de 0 à 11 (il y en a donc 12).

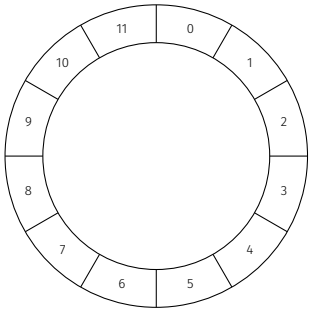


1. On choisit de parcourir les cases en partant de zéro et en se déplaçant à chaque fois de 3 cases, indéfiniment.
   Colorier toutes les case parcourues.
2. Recopier leurs indices (leur numéro) :

   Case parcourues : .....
3. Refaire **1.** et **2.** mais en sautant 4 cases.
   Case parcourues : .....



4. Refaire **1.** et **2.** mais en sautant 5 cases.
   Case parcourues : .....



5. Comment expliquer la différence entre la dernière liste et les deux premières ?

Exercice 52

On parcourt une liste circulaire de longueur 84 comme à l'exercice précédent, en partant de la case d'indice zéro et en sautant 735 cases (et oui cela fait beaucoup) à chaque fois, indéfiniment. La liste sera-t-elle parcourue entièrement ? Si ce n'est pas le cas, donner la liste des cases parcourues.
Justifier les réponses.

Chapitre 4

Logique propositionnelle

1 Notion de proposition

Définition : proposition

Une proposition est un énoncé qui a un sens et pour lequel on peut dire avec certitude qu'il est vrai ou faux. On dit qu'on peut lui associer une *valeur de vérité*. Cette valeur peut se noter VRAI ou FAUX mais on peut aussi choisir de la noter 0 (pour faux) ou 1 (pour vrai).

Exemples

- « $2+2=5$ » est une proposition fausse.
- « $2+2\equiv 1\ [3]$ » est une proposition vraie.

Exercice 53

L'affirmation « Cette affirmation est fausse » est-elle une proposition ?

2 Connecteurs logiques

Définition : négation d'une proposition

L'opérateur de négation se note avec une barre $\bar{}$. C'est un opérateur *unaire*, c'est à dire qu'il s'applique à *une* proposition.
Il est défini par la table de vérité suivante :

P	$\overline{P}$
0	1
1	0

$\overline{P}$ se lit « non P ».

Définition : conjonction de deux propositions

L'opérateur de *conjonction* correspond au `and` de PYTHON, au `And` de VISUAL BASIC, au `&&` de C++.
Il se note $\wedge$, c'est un opérateur *binaire* car il s'applique à deux propositions.
Il est défini par la table de vérité suivante :

P	Q	P ∧ Q
0	0	0
0	1	0
1	0	0
1	1	1

P ∧ Q se lit « P et Q » et n'est vrai que si P est vrai et Q aussi.

Définition : disjonction de deux propositions

L'opérateur de *disjonction* correspond au `or` de PYTHON, au `Or` de VISUAL BASIC, au `||` de C++. Il se note ∨, c'est un opérateur *binaire*. Il est défini par la table de vérité suivante :

P	Q	P ∨ Q
0	0	0
0	1	1
1	0	1
1	1	1

P ∨ Q se lit « P ou Q » et est vrai dès que P est vrai ou Q est vrai.

Définition : équivalence de deux propositions

Il se note ⇔, c'est un opérateur *binaire*. Il est défini par la table de vérité suivante :

P	Q	P ⇔ Q
0	0	1
0	1	0
1	0	0
1	1	1

P ⇔ Q se lit « P équivaut à Q » et n'est vrai que si P et Q ont la même valeur de vérité.

Définition : implication

Il se note ⇒, c'est un opérateur *binaire*. Il est défini par la table de vérité suivante :

P	Q	P ⇒ Q
0	0	1
0	1	1
1	0	0
1	1	1

P ⇒ Q se lit « P implique Q ».

- Quand P est fausse, P ⇒ Q est vraie : « le faux implique n'importe quoi ».
- Quand P est vraie, P ⇒ Q n'est vraie que si Q est aussi vraie : « le vrai n'implique que le vrai ».

Exercice 54

On note P et Q les affirmations suivantes :
P = « Paul aime le foot »
Q = « Paul aime les maths »
Représenter les affirmations suivantes sous forme symbolique en utilisant P, Q et des connecteurs logiques.
- A = « Paul aime le foot mais pas les maths »
- B = « Paul n'aime ni le foot, ni les maths »
- C = « Paul aime le foot ou il aime les maths et pas le foot »
- D = « Paul aime les maths et le foot ou il aime les maths mais pas le foot »

Exercice 55

Donner les valeurs de vérité des propositions suivantes :
- A = $(\pi = 5) \wedge (2 + 3 = 5)$
- B = $(\pi = 5) \vee (2 + 3 = 5)$
- C = $(\pi = 3,14) \Rightarrow (5 + 6 = 11)$
- D = $(\pi = 5) \Rightarrow (2 + 3 = 5)$
- E = $(4 = 5) \Rightarrow$ A
- F = $(5 + 5 = 10) \Leftrightarrow (\pi = 11)$

Exercice 56 (à faire plus tard)

Donner les valeurs de vérité des propositions suivantes :
- A = $(11 > 0) \wedge (3 < 2)$
- B = $(11 > 0) \vee (3 < 2)$
- C = $(3 > 6) \vee (6 > 20)$
- D = $(3 < 2) \Rightarrow (5 = 5)$
- E = $(4 \neq 1) \Rightarrow (4 = 1)$
- F = $(4 < 5) \Leftrightarrow (10 + 1 = 11)$

Méthode : Montrer qu'une proposition est vraie

On peut montrer qu'une proposition composée est vraie en faisant prendre toutes les valeurs de vérités possibles aux propositions qui la compose et en trouvant sa table de vérité :

Montrons que $(P \wedge Q) \Leftrightarrow (Q \wedge P)$ est vraie quelque soient les valeurs de vérité de P et de Q.

P	Q	$P \wedge Q$	$Q \wedge P$	$(P \wedge Q) \Leftrightarrow (Q \wedge P)$
0	0	0	0	1
0	1	0	0	1
1	0	0	0	1
1	1	1	1	1

Exercice 57 : l'implication

Vérifier que la table de vérité de $P \Rightarrow Q$ est la même que celle de $\overline{P} \vee Q$.

Exercice 58 : l'équivalence comme double implication

Vérifier que la table de vérité de $P \Leftrightarrow Q$ est la même que celle de $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$.

Exercice 59 : le ou exclusif

Notons *xor* cet opérateur binaire. P *xor* Q est vraie si (et seulement si) une et une seule des 2 propositions est vraie.

1. Donner la table de vérité de P xor Q.
2. Vérifier que c'est la même que $(P \wedge \overline{Q}) \vee (\overline{P} \wedge Q)$
3. Vérifier que c'est la même que celle de $(P \vee Q) \wedge \overline{(P \wedge Q)}$

Exercice 60 : les lois de De Morgan

1. Montrer que $\overline{P \wedge Q} \Leftrightarrow \overline{P} \vee \overline{Q}$
2. Montrer que $\overline{P \vee Q} \Leftrightarrow (\overline{P} \wedge \overline{Q})$

Propriété : équivalences classiques

Les propositions suivantes sont vraies quelles que soient les valeurs de vérité de P, Q et R.
On dit que ce sont des *tautologies*.

$(P \wedge Q) \Leftrightarrow (Q \wedge P)$	commutativité de $\wedge$
$(P \vee Q) \Leftrightarrow (Q \vee P)$	commutativité de $\vee$
$((P \vee Q) \vee R) \Leftrightarrow (P \vee (Q \vee R))$	associativité de $\vee$
$((P \wedge Q) \wedge R) \Leftrightarrow (P \wedge (Q \wedge R))$	associativité de $\wedge$
$(P \wedge (Q \vee R)) \Leftrightarrow ((P \wedge Q) \vee (P \wedge R))$	distributivité de $\wedge$ sur $\vee$
$(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$	distributivité de $\vee$ sur $\wedge$
$(P \Rightarrow Q) \Leftrightarrow (\overline{P} \vee Q)$	
$\overline{P \wedge Q} \Leftrightarrow (\overline{P} \vee \overline{Q})$	loi de De Morgan
$\overline{P \vee Q} \Leftrightarrow (\overline{P} \wedge \overline{Q})$	loi de De Morgan

Exemple : utilité de le proposition suivante

- « Il viendra mardi et il apportera son PC ou bien il viendra mercredi et il apportera son PC » se simplifie en :
  « Il viendra mardi ou mercredi et il apportera son PC ».
- « On n'a pas : Jean est gentil ou Jean est drôle » peut se réécrire :
  « Jean n'est pas gentil et Jean n'est pas drôle ».

Exercice 61

Simplifier « On n'a pas : Pierre habite Saint Brieuc et Pierre est brun ».

3 Calcul des prédicats

Définitions : quantificateurs, variables, prédicats

Le symbole $\forall$ se lit « pour tout » et s'appelle *quantificateur universel*.
Le symbole $\exists$ se lit « il existe » et s'appelle *quantificateur existentiel*.

Une *variable* est un symbole qui peut prendre plusieurs valeurs.
Un *prédicat* est un énoncé sans valeur de vérité qui contient au moins une variable, et qui devient une proposition en ajoutant un ou des quantificateurs.

Exemples

- « $x < 1$ » est un prédicat comportant une variable x.
- $\exists x \in \mathrm{R},\ x < 1$ est une proposition. Cette proposition est vraie : il existe un nombre réel strictement plus petit que 1 (et même une infinité) : 0 par exemple.
- $\forall x \in \mathbf{R},\ x < 1$ est une autre proposition... fausse ! Tout nombre réel n'est pas strictement plus petit que 1 : 2 par exemple.

Propriété : ordre des quantificateurs

Dans un prédicat à plusieurs variables, quand plusieurs quantificateurs de la même catégorie se suivent, on peut les échanger librement.
On *ne peut pas* échanger un quantificateur $\exists$ et un quantificateur $\forall$.

Exemples

- $\forall x \in \mathbf{R},\ \forall y \in \mathbf{R},\ \exists z \in \mathbf{R}, x + y < z$ est une proposition vraie : pour tous réels x et y on peut prendre z égal à $x + y + 1$.
  On peut échanger les quantificateurs universels : $\forall y \in \mathbf{R},\ \forall x \in \mathbf{R},\ \exists z \in \mathbf{R}, x + y < z$ est équivalent à la proposition précédente.
- $\forall x \in \mathbf{R},\ \exists y \in \mathbf{R}, x < y$ est une proposition vraie mais on ne peut pas échanger les quantificateurs : on obtient : $\exists y \in \mathbf{R},\ \forall x \in \mathbf{R}, x < y$ est fausse : cela voudrait dire qu'il existe un réel y plus grand que tous les autres !

Propriété : négation d'une proposition quantifiée

On obtient la négation d'une proposition quantifiée en changeant les $\exists$ en $\forall$, les $\forall$ en $\exists$ et en changeant le prédicat final par sa négation.

Exemple

On considère la propriété P :

$$\forall n \in \mathbf{N},\ \exists k \in \mathbf{N},\ n = 2k$$

Sa négation est $\overline{\mathrm{P}}$:

$$\exists n \in \mathbf{N},\ \forall k \in \mathbf{N},\ n \neq 2k$$

P est fausse puisqu'elle affirme que tout entier naturel est divisible par 2 !
Sa négation est vraie : elle affirme qu'il existe un entier naturel qui n'est pas divisible par 2 (3 par exemple).

Méthodes : preuve de propositions quantifiées

- Pour prouver qu'une proposition quantifiée par $\forall$ est fausse, il suffit de donner un *contre exemple*.
- Pour prouver qu'une proposition quantifiée par $\exists x$... est vraie, on peut déterminer la valeur de x qui convient.
- Pour prouver qu'une proposition quantifiée par $\forall$ est vraie on a souvent recours à un raisonnement ou au calcul littéral.
- De même pour prouver qu'une proposition quantifiée par $\exists x$... est fausse.

Exemples

- Montrons que $\forall x \in \mathbf{R},\ \exists y \in \mathbf{R},\ 3y+1=x$:
  Soit $x \in \mathbf{R}$ alors $3y+1=x \Leftrightarrow y=\frac{1}{3}(x-1)$. Donc $\frac{1}{3}(x-1)$ convient.
- Montrons que $\forall x \in \mathbf{R},\ \exists y \in \mathbf{R},\ x=y^2$ est fausse :
  Prenons $x=-1$. Il n'existe aucun $y \in \mathbf{R}$ tel que $x=y^2$. En effet d'après la règle des signes, y^2 est obligatoirement positif.

Exercice 62

Vrai ou faux ? Justifier.

- $\forall n \in \mathbf{N},\ \forall p \in \mathbf{N},\ p-n \equiv 0\ [2]$
- $\forall n \in \mathbf{N},\ \exists p \in \mathbf{N},\ p-n \equiv 0\ [2]$
- $\exists n \in \mathbf{N},\ \exists p \in \mathbf{N},\ p-n \equiv 0\ [2]$
- $\exists n \in \mathbf{N},\ \forall p \in \mathbf{N},\ p-n \equiv 0\ [2]$

Exercice 63

Donner les négations des propositions suivantes et dire laquelle est vraie : la proposition ou sa négation.

- $\exists x \in \mathbf{R},\ 3x=2$
- $\forall x \in \mathbf{R},\ x=x+1$
- $\forall x \in \mathbf{R},\ \forall y \in \mathbf{R}, x \leqslant y$
- $\exists x \in \mathbf{R},\ \forall y \in \mathbf{R},\ x^2=y$
- $\forall x \in \mathbf{R},\ \exists y \in \mathbf{R},\ x^2=y$

4 Exercices

Exercice 64

En utilisant les tables de vérités, montrer que, quelles que soient les valeurs de vérité de P et Q, on a

$$\overline{P} \wedge \overline{Q} \Leftrightarrow \overline{P \vee Q}$$

Compléter

P	Q	$\overline{P}$	$\overline{Q}$	$\overline{P} \wedge \overline{Q}$	$P \vee Q$	$\overline{P \vee Q}$

Indiquer les colonnes identiques qui permettent de conclure.

Exercice 65

En utilisant les tables de vérités, montrer que, quelles que soient les valeurs de vérité de P et Q, on a

$$(P \vee Q) \wedge (P \vee \overline{Q}) \Leftrightarrow P$$

Compléter

P	Q	$\overline{P}$	$\overline{Q}$	$P \vee Q$	$P \vee \overline{Q}$	$(P \vee Q) \wedge (P \vee \overline{Q})$

Indiquer les colonnes identiques qui permettent de conclure.

Exercice 66 : lois de De Morgan

En utilisant des tables de vérité, montrer que, quelles que soient les valeurs de vérité de P et Q, on a

$$\overline{P \vee Q} = \overline{P} \wedge \overline{Q}$$

De même montrer que

$$\overline{P \wedge Q} = \overline{P} \vee \overline{Q}$$

Exercice 67 : on peut retrouver tous les opérateurs à partir du nor

Pour toutes propositions A et B on définit l'opération « nor » , notée $\downarrow$ par :

$$A \downarrow B \iff \overline{A \vee B}$$

Cette opération est dite *universelle* car elle permet de retrouver toutes les autres opérations.

1. Montrer que $A \downarrow A \iff \overline{A}$ (on peut donc retrouver l'opération « non »).
2. En déduire que l'on peut retrouver l'opération « et » ainsi :

$$(A \downarrow B) \downarrow (A \downarrow B) = A \vee B$$

3. Comment à partir de A, B et $\downarrow$ obtenir $A \wedge B$ (penser aux lois de De Morgan) ?

Exercice 68

Sans chercher à démontrer quoi que ce soit, donner les négations des propositions suivantes

1. $\forall x \in \mathbf{R}, \forall y \in \mathbf{R}, \exists z \in \mathbf{R}, x < z < y$
2. $\exists x \in \mathbf{R}, \exists y \in \mathbf{R}, x + y > 3$
3. $\forall n \in \mathbf{N}^*, \exists p \in \mathbf{N}^*$ n divise p ou p divise n.

Exercice 69

1. A : $\forall n \in \mathbf{N}$ 3 divise n ou 2 divise n
   Montrer que A est fausse
2. B : $\exists n \in \mathbf{N}$, 3 divise n et 4 divise n
   Montrer que B est vraie
3. C : « Quand on prend trois nombres entiers qui se suivent, leur somme est toujours un multiple de 3 » .
   Montrer que C est vraie.
4. D : « Quand on prend quatre nombres entiers qui se suivent, leur somme est toujours un multiple de 4 » .
   Montrer que D est fausse.
5. E : « Il existe deux entiers k et n plus grands que 1 tels que k divise à la fois n et n+1.
   Montrer que E est fausse.

Chapitre 5

Matrices

1 Notion de matrice

Définition : matrice

Une matrice A peut être vue comme « un tableau de nombres ».
Supposons qu'elle comporte n lignes et p colonnes (n et p sont des entiers plus grands que 1), on la note ainsi

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{pmatrix}$$

L'élément qui se situe à la i^{e} ligne et à la j^{e} colonne est noté a_{ij}. On l'appelle également *coefficient*.

Attention : les indices des lignes et des colonnes commencent à 1 (et non à zéro comme dans la plupart des langages informatiques).

Pour résumer l'écriture précédente on écrit

$$\mathrm{A} = (a_{ij})_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant p}}$$

On dit aussi que A est une matrice $n \times p$. Si $n = p$ on dit que A est une *matrice carrée d'ordre n*.

Exemples

- $\mathrm{B} = \begin{pmatrix} 1 & 2 & 4 \\ -3 & 5 & 0 \end{pmatrix}$ est une matrice à 2 lignes et 3 colonnes. On a $b_{21} = -3$.
- $\mathrm{C} = \begin{pmatrix} 2,4 & 7 & 4 & -1 \\ -3 & 5 & 10,1 & 1 \\ 0,01 & 3 & 12 & 100 \end{pmatrix}$ est une matrice à 3 lignes et 4 colonnes. On a $c_{33} = 12$.
- $\mathrm{D} = \begin{pmatrix} 4 & 2 \\ 2 & 8 \end{pmatrix}$ est une matrice carrée d'ordre 2.

Exercice 70

On considère $E = \begin{pmatrix} -4 & 7,6 & 4 & -1 & 12 \\ 8 & -3 & 5,7 & 101 & 1 \\ 12 & 0,01 & 3 & 12 & 1 \end{pmatrix}$.

Donne les valeurs de e_{12}, e_{21}, e_{35} et e_{24}.

Le script PYTHON suivant permet de générer une matrice $n \times p$ avec des coefficients entiers aléatoires compris entre -100 et 100.

Python

```
from random import randint

n = int(input("Entrez le nombre de lignes : "))
p = int(input("Entrez le nombre de colonnes : "))

matrice = []  # une matrice est une liste de lignes

for i in range(n):  # il y a n lignes
    ligne = []  # on construit une ligne vide
    for j in range(p):  # il y a p colonnes
        ligne.append(randint(-100, 100))  # on remplit la ligne
        ↪ aléatoirement
    matrice.append(ligne)  # on ajoute la ligne à la liste de lignes
```

Exercice 71

1. Écris complètement la matrice suivante : $M = (m_{ij})_{\substack{1 \leqslant i \leqslant 3 \\ 1 \leqslant j \leqslant 5}}$ où $m_{ij} = i$ si i=j et 0 sinon.
2. Écris complètement la matrice suivante : $M = (m_{ij})_{\substack{1 \leqslant i \leqslant 4 \\ 1 \leqslant j \leqslant 5}}$ où $m_{ij} = 0$ si i<j et 1 sinon.
3. Écris complètement la matrice suivante : $M = (m_{ij})_{\substack{1 \leqslant i \leqslant 3 \\ 1 \leqslant j \leqslant 3}}$ où $m_{ij} = 1$ si $i + j$ est pair et 0 sinon.
4. BONUS : écris des programmes PYTHON qui génèrent ces matrices.

Définitions : Matrices nulles et identités

- Une matrice dont tous les coefficients sont nuls est dite *nulle* (c'est « un tableau de zéros »);
- la matrice *carrée d'ordre n* dont tous les éléments sont nuls sauf ceux de la *diagonale* (c'est-à-dire ceux qui s'écrivent a_{ii}) qui valent 1 s'appelle *la matrice identité d'ordre n* et se note I_n.

Exemple

La matrice identité I_3 est $I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

2 Opérations sur les matrices

Définition : addition

Soient A et B deux matrices $n \times p$, on note $A+B$ la matrice $n \times p$ obtenu en ajoutant les coefficients correspondants de A et de B :

$$\begin{pmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{np} \end{pmatrix} + \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & \cdots & a_{1p}+b_{1p} \\ \vdots & \ddots & \vdots \\ a_{n1}+n_{n1} & \cdots & a_{np}+b_{np} \end{pmatrix}$$

Exemple

Prenons $A = \begin{pmatrix} 3 & 1 \\ 4 & 7 \\ -2 & 8 \end{pmatrix}$ et $B = \begin{pmatrix} 2 & 5 \\ 1 & -3 \\ -5 & 9 \end{pmatrix}$, alors $A+B = \begin{pmatrix} 5 & 6 \\ 5 & 4 \\ -7 & 17 \end{pmatrix}$.

Remarque

Attention : on ne peut ajouter deux matrices que si elles ont les mêmes dimensions (c'est-à-dire même nombre de lignes et même nombre de colonnes).

Définition : multiplication par un réel

Soient A une matrice $n \times p$ et k un nombre réel, on note kA la matrice $n \times p$ obtenue en multipliant chaque coefficient de A par k :

$$k\begin{pmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{np} \end{pmatrix} = \begin{pmatrix} k \times a_{11} & \cdots & k \times a_{1p} \\ \vdots & \ddots & \vdots \\ k \times a_{n1} & \cdots & k \times a_{np} \end{pmatrix}$$

Exemple

Prenons $A = \begin{pmatrix} 8 & 3 & 1 \\ 4 & 7 & 2 \\ -2 & 1 & 8 \end{pmatrix}$ et $k = 5$, alors on obtient que $5A = \begin{pmatrix} 40 & 15 & 5 \\ 20 & 35 & 10 \\ -10 & 5 & 40 \end{pmatrix}$.

La propriété suivante énonce quelques résultats utiles pour calculer.

Propriété : règles de calcul

Soient A, B et C trois matrices de mêmes dimensions et k et k' 2 réels.

- $A+B = B+A$
- $(A+B)+C = A+(B+C)$
- $k(A+B) = kA+kB$
- $(k+k')A = kA+k'A$

Exercice 72

On pose $A = \begin{pmatrix} 1 & 2 \\ 3 & -4 \end{pmatrix}$, $B = \begin{pmatrix} 11 & 10 \\ -9 & 7 \end{pmatrix}$ et $C = \begin{pmatrix} -3 & -2 \\ 5 & -5 \end{pmatrix}$.

Montrer que $B - 2A + 3C$ est une matrice nulle.

Définition : multiplication de deux matrices

Soient A une matrice $n \times p$ et B une matrice $p \times q$ (le nombre de colonnes de la 1re est égal au nombre de lignes de la 2e) alors il est possible de définir la matrice $C = A \times B$, produit de A par B. C'est une matrice $n \times q$ dont les coefficients sont ainsi :

$$\begin{pmatrix} b_{11} & \cdots & b_{1j} & \cdots & a_{1q} \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ b_{k1} & \cdots & b_{kj} & \cdots & b_{kq} \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ b_{p1} & \cdots & b_{pj} & \cdots & b_{pq} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & \cdots & \cdots & \cdots & a_{1p} \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ a_{i1} & \cdots & a_{ik} & \cdots & a_{ip} \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{np} \end{pmatrix} \quad \begin{pmatrix} c_{11} & \cdots & \cdots & \cdots & c_{1q} \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ \vdots & \cdots & c_{ij} & \cdots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ c_{n1} & \cdots & \cdots & \cdots & c_{nq} \end{pmatrix}$$

$$c_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \ldots + a_{ip} \times b_{pj}$$

Exemple

Prenons $A = \begin{pmatrix} 1 & 3 & -2 \\ 5 & -3 & 4 \end{pmatrix}$ et $B = \begin{pmatrix} -5 & 1 & 0 & 2 \\ 2 & 1 & -1 & -8 \\ 3 & 4 & 0 & 9 \end{pmatrix}$ alors A est une matrice 2×3, B est une matrice 3×4 donc il est possible de définir la matrice $C = A \times B$, ce sera une matrice 2×4.

$$\begin{pmatrix} -5 & 1 & 0 & 2 \\ 2 & 1 & -1 & -8 \\ 3 & 4 & 0 & 9 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 3 & -2 \\ 5 & -3 & 4 \end{pmatrix} \quad \begin{pmatrix} -5 & -4 & -3 & -40 \\ -19 & 18 & 3 & 70 \end{pmatrix}$$

Par exemple, pour calculer c_{33}, on fait $5 \times 0 + (-3) \times (-1) + 4 \times 0 = 3$.

Remarque

Attention :

- on ne peut multiplier A par B que si le nombre de colonnes de A est égal au nombre de lignes de B ;
- ce n'est pas parce qu'on peut calculer $A \times B$ qu'on peut calculer $B \times A$: les matrices de l'exemple précédent ne permettent pas de calculer $B \times A$ car le nombre de colonnes de B n'est pas égal

au nombre de lignes de A :

$$\begin{pmatrix} 1 & 3 & -2 \\ 5 & -3 & 4 \end{pmatrix}$$

$$\begin{pmatrix} -5 & 1 & 0 & 2 \\ 2 & 1 & -1 & -8 \\ 3 & 4 & 0 & 9 \end{pmatrix} \quad \text{Impossible}$$

- pour pouvoir calculer $A \times B$ et $B \times A$ il faut que ces deux matrices soient carrées d'ordre n et *en général* on n'a pas $A \times B = B \times A$.

Exercice 73

- On pose $A = \begin{pmatrix} 1 & 2 \\ 3 & 0 \end{pmatrix}$ et $B = \begin{pmatrix} 5 & 2 \\ 0 & 3 \end{pmatrix}$.

  Calculer AB et BA.
- Recommencer avec $A = \begin{pmatrix} -4 & 6 \\ -3 & 5 \end{pmatrix}$ et $B = \begin{pmatrix} 8 & -10 \\ 5 & -7 \end{pmatrix}$.

Propriétés de calcul

A, B et C sont des matrices.
Lorsque les opérations sont possibles (bonnes dimensions des matrices) on a :

- $A(BC) = (AB)C$;
- $(A + B)C = AC + BC$;
- $A(B + C) = AB + AC$.

Soit k un nombre réel alors on a également $A \times kB = kAB$.

Si A est *carrée d'ordre n* on a

- $AI_n = I_nA = A$ où I_n est la matrice identité d'ordre n.
- $A \times 0 = 0 \times A = 0$ en notant 0 la matrice carrée d'ordre n nulle.

3 Exemple concret d'utilisation

Imaginons une école qui forme des ingénieurs en informatique, avec seulement 3 matières. Trois élèves de première année ont obtenu les résultats suivants :

Résultats pour le premier trimestre

	Maths	Physique	Info
Adam	12	8	16
Bertrand	18	14	12
Charles	5	20	15

Résultats pour le deuxième trimestre

	Maths	Physique	Info
Adam	10	10	14
Bertrand	18	12	14
Charles	7	14	17

Ces deux tableaux peuvent s'écrire matriciellement $S_1 = \begin{pmatrix} 12 & 8 & 16 \\ 18 & 14 & 12 \\ 5 & 20 & 15 \end{pmatrix}$ et $S_2 = \begin{pmatrix} 10 & 10 & 14 \\ 18 & 12 & 14 \\ 7 & 14 & 17 \end{pmatrix}$

Pour calculer les moyennes mensuelles des élèves « en une fois » on peut définir $M = 0,5(S_1 + S_2)$:

$$M = 0,5 \times \left[\begin{pmatrix} 12 & 8 & 16 \\ 18 & 14 & 12 \\ 5 & 20 & 15 \end{pmatrix} + \begin{pmatrix} 10 & 10 & 14 \\ 18 & 12 & 14 \\ 7 & 14 & 17 \end{pmatrix} \right]$$

$$M = 0,5 \times \begin{pmatrix} 22 & 18 & 30 \\ 36 & 16 & 26 \\ 12 & 34 & 32 \end{pmatrix}$$

$$M = \begin{pmatrix} 11 & 9 & 15 \\ 18 & 8 & 13 \\ 6 & 17 & 16 \end{pmatrix}$$

Le coefficient des mathématiques est 1, celui de la physique est 2 et celui de l'informatique est 5.
Pour passer en deuxième année, il faut un total de points supérieur ou égal à 120.

Pour faire « d'un coup » le total des points, on peut considérer la matrice de coefficients $C = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$.

Les points des élèves sont donnés par la matrice

$$P = MC$$

$$P = \begin{pmatrix} 11 & 9 & 15 \\ 18 & 8 & 13 \\ 6 & 17 & 16 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$$

$$P = \begin{pmatrix} 104 \\ 109 \\ 120 \end{pmatrix}$$

Ainsi seul Charles est admis à passer en 2^e année.

4 Matrices inversibles et systèmes

Définition et propriété : Matrice inversible, inverse d'une matrice

Soit A une matrice **carrée d'ordre** n. S'il existe une matrice B d'ordre n telle que

$$AB = I_n \qquad \text{ou} \qquad BA = I_n$$

alors automatiquement les deux égalités sont vérifiées, B est nécessairement *unique* et on dit alors que B *est l'inverse de* A. De manière symétrique A est également l'inverse de B si bien qu'on dit que A et B sont inverses l'une de l'autre.
On note ceci $A = B^{-1}$ ou, ce qui revient au même, $B = A^{-1}$.

Exemple

$A = \begin{pmatrix} -1 & 2 \\ -2 & 3 \end{pmatrix}$ et $B = \begin{pmatrix} 3 & -2 \\ 2 & -1 \end{pmatrix}$ sont inverses l'une de l'autre :

$$\begin{pmatrix} 3 & -2 \\ 2 & -1 \end{pmatrix}$$
$$\begin{pmatrix} -1 & 2 \\ -2 & 3 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Exercice 74

Montrer que $A = \begin{pmatrix} 3 & -2 & 1 \\ -2 & 2 & -1 \\ 1 & -1 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ sont inverses.

Remarque

Il existe des matrices non inversibles, par exemple $\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$.

Méthode : résoudre des systèmes avec des matrices

On considère un *système de n équations à n inconnues* :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

On connaît tous les nombres a_{ij} et tous les b_i, et on veut trouver les valeurs des inconnues x_i.

Ce système peut se réécrire de manière matricielle :

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Ou encore :

$$AX = B$$

Si la matrice A est inversible (en pratique ce sera toujours le cas parce qu'on nous donnera sa matrice inverse ou bien parce qu'on l'aura déterminée à l'aide de la calculatrice) alors, on peut reprendre l'égalité précédente et écrire : $A^{-1}AX = A^{-1}B$, ce qui donne $I_nX = A^{-1}B$. En définitive on a

$$X = A^{-1}B$$

Ainsi pour trouver les valeurs des inconnues x_i, on effectue simplement le produit matriciel $A^{-1}B$: chacune de ses lignes nous donne la valeur du x_i correspondant.

Remarque

Pour savoir comment utiliser la calculatrice, regarder ici :

- modèles CASIO `https://youtu.be/yjvQx13Vhlk`
- modèles TEXAS INSTRUMENT `https://youtu.be/rxDxBnIwaGo`

Exemple

On considère le système suivant : $\begin{cases} 2x + 5y + 2z = 1 \\ 5x - 3y - 2z = 2 \\ -x + 2y + z = -3 \end{cases}$

Il peut se réécrire de manière matricielle :

$$\begin{pmatrix} 2 & 5 & 2 \\ 5 & -3 & -2 \\ -1 & 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$$

Appelons A la matrice carrée du membre de gauche. On détermine que A est inversible avec la calculatrice et que son inverse est

$$A^{-1} = \begin{pmatrix} 1 & -1 & -4 \\ -3 & 4 & 14 \\ 7 & -9 & -31 \end{pmatrix}$$

On a donc

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & -1 & -4 \\ -3 & 4 & 14 \\ 7 & -9 & -31 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$$

C'est à dire, en effectuant le produit dans le membre de droite

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 11 \\ -37 \\ 82 \end{pmatrix}$$

On a donc résolu le système : $\begin{cases} x = 11 \\ y = -37 \\ z = 82 \end{cases}$

Exercice 75

1. Effectue le produit suivant : $\begin{pmatrix} 1 & 2 \\ -3 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$.
2. Á l'aide de la calculatrice détermine l'inverse de la matrice $\begin{pmatrix} 1 & 2 \\ -3 & 3 \end{pmatrix}$.
3. Résous le système suivant : $\begin{cases} x + 2y = 15 \\ -3x + 3y = -6 \end{cases}$

5 Exercices

Exercice 76

Dans un parc d'une ville, deux marchands ambulants vendent des beignets, des crêpes et des gaufres. On a noté les ventes de chacun pour samedi et dimanche derniers.

Marchand 1

	beignets	crêpes	gaufres
samedi	20	36	12
dimanche	26	40	18

Marchand 2

	beignets	crêpes	gaufres
samedi	30	40	22
dimanche	30	48	38

On peut retenir l'information donnée par un tableau en conservant uniquement les nombres disposés de la même façon. On représente le 1er tableau par la matrice A :

$$\mathrm{A} = \begin{pmatrix} 20 & 36 & 12 \\ 26 & 40 & 18 \end{pmatrix}$$

1. Donner la matrice B représentant le deuxième tableau.
2. Que valent a_{12}, a_{11}, a_{23} et b_{11} ?
3. Calculer A + B et donner la signification de la matrice.

4. Calculer A − B et donner la signification de la matrice.

5. Samedi et dimanche prochains, weekend de fête, on prévoit que les ventes vont augmenter de 50%.
   Par quel nombre k faut-il multiplier chacune des ventes du 1er marchand ? Écrire la matrice kA.
   Donner la matrice kB correspondant aux ventes du 2e marchand.

6. Un beignet est vendu 2 euros, une crêpe 1 euro et une gaufre 1,50 euro.
   On note V la matrice des prix de vente

$$V = \begin{pmatrix} 2 \\ 1 \\ 1,5 \end{pmatrix}$$

   Quelle opération matricielle donne le montant des ventes par jour pour le 1er marchand ? Pour le 2e ?

7. Les deux marchands travaillent pour le compte du même patron, qui leur demande de calculer les coûts d'achats et les revenus pour chaque jour. Le coût d'achat d'un beignet est 0,40 euro, d'une crêpe 0,25 euro, d'une gaufre 0,30 euro. On note T la matrice donnant prix d'achat et prix de vente par catégorie

$$T = \begin{pmatrix} 0,4 & 2 \\ 0,25 & 1 \\ 0,3 & 1,5 \end{pmatrix}$$

   Quelle opération matricielle permet le calcul des coûts d'achat et revenus par jour pour le 1er marchand ?
   Calculer, de même, les coûts d'achats et les revenus par jour pour le 2e marchand puis, globalement, pour le patron.

Exercice 77

$$A = \begin{pmatrix} 1 & 0 & -2 \\ 2 & 3 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 3 & -1 & -2 \end{pmatrix} \text{ et } C = \begin{pmatrix} 3 & 0 & -2 & 0 \\ -2 & 1 & 1 & 1 \\ 1 & -1 & 0 & 3 \end{pmatrix}.$$

1. Calculer $A \times B$, puis $(A \times B) \times C$.

2. Calculer $B \times C$, puis $A \times (B \times C)$.

3. Pouvait-on prévoir ce résultat ?

Exercice 78

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix}, B = \begin{pmatrix} -3 & -6 \\ 1 & 2 \end{pmatrix}, C = \begin{pmatrix} 1 & 1 & 2 \\ 2 & 2 & 4 \\ 3 & 3 & 6 \end{pmatrix} \text{ et } D = \begin{pmatrix} 1 & 2 & -6 \\ 1 & 2 & -6 \\ 1 & -2 & 6 \end{pmatrix}.$$

1. Calculer le produit $A \times B$.

2. Calculer le produit $C \times D$.

3. Que peut-on en conclure ?

Exercice 79

$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.

Calculer A^5.

Exercice 80 : Calculs à la main

On considère les matrice $A = \begin{pmatrix} -5 & 2 \\ -3 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & -2 \\ 3 & -5 \end{pmatrix}$.

1. Montrer *à la main* que A et B sont inverses.
2. On considère le système (S) suivant :

$$\begin{cases} -5x + 2y &= 7 \\ -3x + y &= 8 \end{cases}$$

Montrer que ce système peut se réécrire matriciellement

$$AX = Y$$

et préciser X et Y

3. En déduire *à la main* les solutions du système (S).

Exercice 81 : À la calculatrice

On considère les matrice $A = \begin{pmatrix} 8 & 11 & 3 \\ 4 & 7 & 2 \\ 1 & 3 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 5 & -4 \\ 5 & -13 & 12 \end{pmatrix}$.

1. Comment avec la calculatrice vérifie-t-on que A et B sont inverses ?
2. On considère le système (S) suivant :

$$\begin{cases} 8x + 11y + 3z &= 1 \\ 4x + 7y + 2z &= 2 \\ x + 3y + z &= 3 \end{cases}$$

Montrer que ce système peut se réécrire matriciellement

$$AX = Y$$

et préciser X et Y

3. En déduire *à la main* les solutions du système (S).

Exercice 82

À la papeterie :

- 3 stylos, 2 cahiers et 4 gommes coûtent 6,30€ ;
- 5 stylos, 7 cahiers et 1 gomme coûtent 15€ ;
- 10 stylos, 1 cahier et 6 gommes coûtent 6€.

À l'aide de la calculatrice et en expliquant la démarche, déterminer le prix de chaque article.

Exercice 83

Une société produit trois types de fibres optiques à partir de silice, forme naturelle du dioxyde de silicium (SiO_2) qui entre dans la composition de nombreux minéraux. Elle produit :

- x pièces du type A, dont le débit supporté vaut 1 gigabit par seconde ;
- y pièces du type B, dont le débit supporté vaut 10 gigabits par seconde ;
- z pièces du type C, dont le débit supporté vaut 100 gigabits par seconde.

Pour une pièce, la masse de silice utilisée et le temps de production de chacun de ces types de fibres sont récapitulés dans le tableau suivant.

Type de fibre	A	B	C
Masse de silice en kg (par pièce)	3	4	7
Temps de production en h (par pièce)	2	3	5

La société modélise cette fabrication afin d'envisager différents scénarios sur une période donnée. Pour cette période, on note N le nombre total de pièces produites, S la masse totale en kg de silice utilisée et H le temps total de production exprimé en heure.

1. Justifier le fait que x, y, z vérifient le système $\begin{cases} x+y+z &= \mathrm{N} \\ 3x+4y+7z &= \mathrm{S} \\ 2x+3y+5z &= \mathrm{H} \end{cases}$.

2. On considère les matrices colonnes $\mathrm{X} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ et $\mathrm{Y} = \begin{pmatrix} \mathrm{N} \\ \mathrm{S} \\ \mathrm{H} \end{pmatrix}$. Déterminer la matrice carrée M qui traduit le système ci-dessus par l'équation matricielle $\mathrm{M} \times \mathrm{X} = \mathrm{Y}$.

3. Calculer Y lorsque $\mathrm{X} = \begin{pmatrix} 20 \\ 10 \\ 30 \end{pmatrix}$. Interpréter les résultats obtenus dans le contexte de l'exercice.

4. On considère la matrice carrée $\mathrm{P} = \begin{pmatrix} 1 & 2 & -3 \\ 1 & -3 & 4 \\ -1 & 1 & -1 \end{pmatrix}$.

a. Calculer le produit matriciel $\mathrm{P} \times \mathrm{M}$.

b. Montrer que si $\mathrm{M} \times \mathrm{X} = \mathrm{Y}$, alors $\mathrm{X} = \mathrm{P} \times \mathrm{Y}$.

c. Pour une période donnée, l'entreprise dispose de 94 kg de silice et de 67 heures de production. Elle souhaite fabriquer 21 pièces de fibres.
Combien de pièces de chaque type peut-elle fabriquer ?

Chapitre 6

Algèbres de Boole

1 Définition d'une algèbre de Boole

Définition

Une *algèbre de Boole*, c'est un ensemble E muni de :

- deux lois *binaires* notées + et $\times$;
- une loi *unaire* qui à a associe $\overline{a}$;
- deux éléments particuliers notés 0 et 1;

et telle que , pour tous éléments a, b et c de E :

- Les deux lois binaires sont *commutatives* : $a+b=b+a$ et $ab=ba$
- elles sont aussi associatives : $a+(b+c)=(a+b)+c$ et $a(bc)=(ab)c$
- $\times$ se distribue sur $+$: $a(b+c)=ab+ac$
- + se distribue sur $\times$: $\boxed{a+bc=(a+b)(a+c)}$
  c'est la seule propriété contre-intuitive, on la notera $(*)$
- 0 est neutre pour $+$: $0+a=a$
- 0 est absorbant pour $\times$: $0a=0$
- 1 est neutre pour $\times$: $1a=a$
- 1 est absorbant pour $+$: $1+a=1$
- on a $a+\overline{a}=1$ et $a\overline{a}=0$

Exemples

- L'ensemble $\{0;1\}$ muni de $\vee$, $\wedge$ et $\overline{\ }$ est la plus simple des algèbres de Boole.
- Si on pose que deux propositions sont égales quand elles sont équivalentes, alors l'ensemble des propositions muni de $\vee$, $\wedge$ et $\overline{\ }$ est une algèbre de Boole, 0 étant la proposition fausse et 1 la proposition vraie.
- Comme on le verra au prochain chapitre, lorsqu'on se donne un ensemble E et que l'on considère $\mathcal{P}(E)$, l'ensemble de ses parties, muni des opérations $\cap$ (intersection, joue le rôle de $\times$) et $\cup$ (union, joue le rôle de +), alors P(E) est une algèbre de Boole, 0 étant l'*ensemble vide* et 1 étant E lui-même.

Exercice 84

Montrer que :

1. $(a+b)(\overline{a}+c)(\overline{b}+\overline{c}) = a\overline{b}c + \overline{a}b\overline{c}$
2. $\overline{a}\,\overline{b}\,\overline{c}\,\overline{d} + a\overline{b}\,\overline{c}\,\overline{d} + \overline{a}b\overline{c}\,\overline{d} + ab\overline{c}\,\overline{d} = \overline{b}\,\overline{c}$
3. $ab + \overline{a}\,\overline{b} + \overline{a}b = \overline{a} + b$

Propriétés

- Dans une algèbre de Boole, pas besoin de multiplication par un entier car pour tout élément a :

$$a + a = a$$

  Donc $2a = a$ et par suite pour tout entier naturel non nul n : $na = a$.
- Pas besoin de puissances non plus car pour tout élément a :

$$aa = a$$

  Donc $a^2 = a$ et par suite pour tout entier naturel non nul n : $a^n = a$.

Preuve :
Soit a un élément de E :

- $a = a + 0$
  $= a + a\overline{a}$ et on utilise $(*)$
  $= (a+a)(a+\overline{a})$
  $= (a+a)1$
  $= a + a$

- $a = a \times 1$
  $= a(a+\overline{a})$
  $= aa + a\overline{a}$
  $= aa + 0$
  $= aa$

Propriété

Si 2 éléments x et y de E sont tels que $xy = 0$ et $x + y = 1$, alors $x = \overline{y}$.

Preuve :

$$\begin{aligned} x+y=1 &\Rightarrow (x+y)\overline{y} = \overline{y} \\ &\Rightarrow x\overline{y} + y\overline{y} = \overline{y} \\ &\Rightarrow x\overline{y} + 0 = \overline{y} \end{aligned}$$

$$\Rightarrow \quad x\overline{y} + xy = \overline{y}$$
$$\Rightarrow \quad x(\overline{y} + y) = \overline{y}$$
$$\Rightarrow \quad x \times 1 = \overline{y}$$
$$\Rightarrow \quad x = \overline{y}$$

Propriété : lois de De Morgan

Quels que soient les éléments de x et y de E on a

$$\overline{x \times y} = \overline{x} + \overline{y} \qquad \text{et} \qquad \overline{x + y} = \overline{x} \times \overline{y}$$

Preuve de la première égalité.
Si on pose $A = xy$ et $B = \overline{x} + \overline{y}$, alors on doit montrer que $\overline{A} = B$.
On utilise la propriété précédente et on montre donc que $A + B = 1$ et que $AB = 0$:

$B + A = \overline{x} + \overline{y} + xy$ et on utilise $(*)$
$= (\overline{x} + \overline{y} + x)(\overline{x} + \overline{y} + y)$
$= (1 + \overline{y})(1 + \overline{x})$
$= 1 \times 1$
$= 1$

$AB = (\overline{x} + \overline{y})xy$
$= \overline{x}xy + \overline{y}xy$
$= 0y + 0x$
$= 0 + 0$
$= 0$

Exercice 85

Montrer la deuxième loi en faisant la même chose.

2 Diagrammes de Karnaugh

Cette méthode a été développée par Maurice Karnaugh en 1953. Elle fait appel à des tableaux qui représentent des expressions booléennes.
On colorie les cases qui correspondent aux différents termes de l'expression. À la fin on peut reconnaître et lire une expression simplifiée.

Propriété

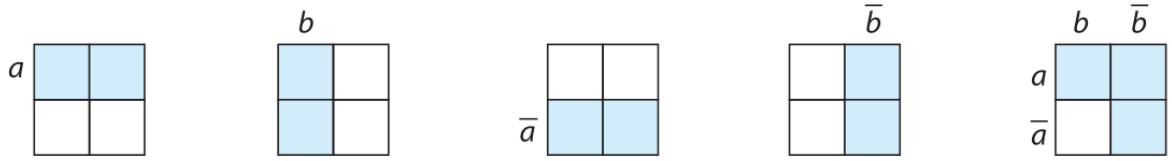
Un produit correspond à une *intersection*, une somme à une *union*.

2.1 Avec 2 variables

Le diagramme est un carré dont les quatre cases correspondent aux quatre produits ab, $a\overline{b}$, $\overline{a}\,b$, $\overline{a}\,\overline{b}$.

ab	$a\overline{b}$
$\overline{a}\,b$	$\overline{a}\,\overline{b}$

Ci dessous on représente a, b, $\overline{a}$, $\overline{b}$ et une dernière expression.



Cette dernière peut se lire $a + \overline{b}$ ou bien $ab + a\overline{b} + \overline{a}\,\overline{b}$.

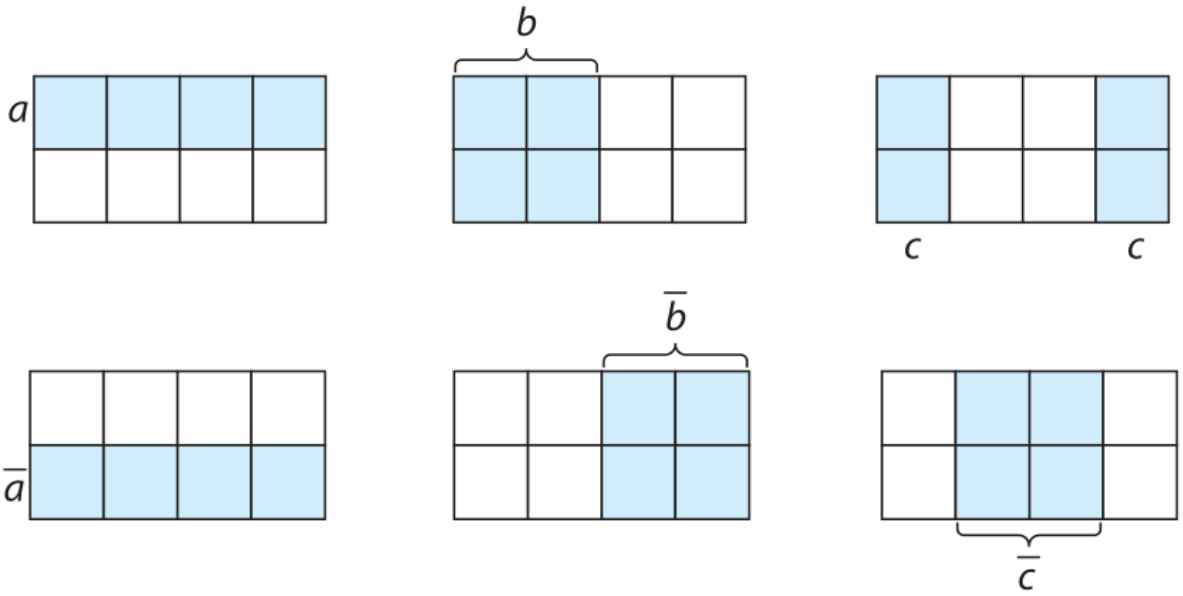
Exercice 86

Propose une troisième interprétation du dernier diagramme.

2.2 Avec 3 variables

Le diagramme est un rectangle dont les huit cases correspondent aux produits suivants.

$$\begin{array}{cccc} abc & ab\overline{c} & a\overline{b}\,\overline{c} & a\overline{b}\,c \\ \overline{a}\,bc & \overline{a}\,b\overline{c} & \overline{a}\,\overline{b}\,\overline{c} & \overline{a}\,\overline{b}\,c \end{array}$$



Exercice 87

Faire les tables de Karnaugh des expressions ab, $a\overline{b}$, ac, $a\overline{c}$, $\overline{a}\,b$, $\overline{a}\,\overline{b}$, $\overline{a}\,c$, $\overline{a}\,\overline{c}$, bc, $\overline{b}\,c$, $b\overline{c}$, $\overline{b}\,\overline{c}$.

3 Exercices

Exercice 88

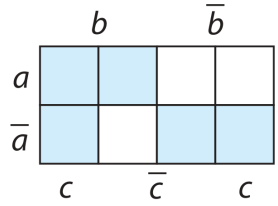
$\mathcal{B}$ est une algèbre de Boole.

1. Montrer par le calcul que $\forall a \in \mathcal{B}, \forall b \in \mathcal{B},\ a + ab = a$;
2. Montrer par le calcul que $\forall a \in \mathcal{B}, \forall b \in \mathcal{B},\ a(a + b) = a$.
3. Montrer par le calcul que $\forall a \in \mathcal{B}, \forall b \in \mathcal{B},\ a + \overline{a}\,b = a + b$.

Vérifier ces trois égalités avec des tables de Karnaugh.

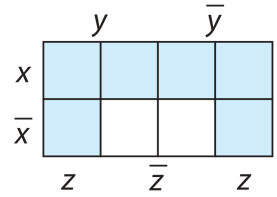
Exercice 89

Écrire de deux façons possibles l'expression booléenne représentée par le tableau de Karnaugh suivant.



Exercice 90

Écrire l'expression booléenne représentée par le tableau de Karnaugh suivant sous la forme d'une somme de deux variables booléennes prises parmi x, y, z, $\overline{x}$, $\overline{y}$ et $\overline{z}$.



Exercice 91

x,y et z sont trois élément d'une algèbre booléenne B.

Écrire l'expression $y\overline{(xy+z)}$ sous la forme d'un produit de trois variables booléennes prises parmi x, y, z, $\overline{x}$, $\overline{y}$ et $\overline{z}$.

Exercice 92

a,b et c sont trois élément d'une algèbre booléenne B.

1. Écrire l'expression $(a+\overline{b}\,c)(b+\overline{c}\,)$ sous la forme d'une somme de deux produits de deux variables booléennes prises parmi a, b, c, $\overline{a}$, $\overline{b}$ et $\overline{c}$.
2. Représenter ce résultat dans une table de Karnaugh et en déduire une nouvelle expression.

Exercice 93

B est une algèbre booléenne. On définit l'opération « nor », notée $\downarrow$ par :

$$\forall a \in \mathcal{B}, \forall b \in \mathcal{B},\ a \downarrow b = \overline{a+b}$$

Cette opération est dite *universelle* car elle permet de retrouver toutes les autres opérations.

1. Montrer que $\forall a \in \mathcal{B}\ a \downarrow a = \overline{a}$.
2. En déduire que
$$\forall a \in \mathcal{B}, \forall b \in \mathcal{B},\ (a \downarrow b) \downarrow (a \downarrow b) = a+b$$
3. Comment à partir de a, b et $\downarrow$ obtenir ab ?

Exercice 94 : Polynésie juin 2018

Une société de fabrication et d'installation de fibre optique a besoin de recruter un informaticien, femme ou homme. La direction des ressources humaines considère qu'une candidature est

recevable lorsqu'elle satisfait à l'une au moins des conditions suivantes :

- le candidat est âgé de 25 ans ou moins et est titulaire du BTS SIO ;
- le candidat est âgé de 25 ans ou moins, n'est pas titulaire du BTS SIO et possède de l'expérience ;
- le candidat est âgé de strictement plus de 25 ans et est titulaire du BTS SIO ;

On définit les variables booléennes a, b, c de la façon suivante :

- $a = 1$ si le candidat est âgé de strictement plus de 25 ans, $a = 0$ sinon ;
- $b = 1$ si le candidat est titulaire d'un BTS SIO, $b = 0$ sinon ;
- $c = 1$ si le candidat a de l'expérience, $c = 0$ sinon.

1. Écrire une expression booléenne E traduisant qu'une candidature est recevable, à l'aide des variables booléennes a, b, c.
2. À l'aide d'un tableau de Karnaugh, déterminer une écriture simplifiée de E sous la forme d'une somme de deux termes. En déduire une interprétation simplifiée des conditions pour qu'une candidature soit recevable.
3. Une candidate a 21 ans, aucune expérience, mais est titulaire du BTS SIO. Remplit-elle les critères de recrutement ?
4. Donner une expression simple de $\overline{\text{E}}$.

Exercice 95 : Polynésie mai 2017

Cinq joueurs, notés A, B, C, D et E, jouent régulièrement à un jeu en ligne.
Chaque partie de ce jeu oppose deux adversaires.

1. Dans cette question, on note $\text{J} = \{\text{A}, \text{B}, \text{C}, \text{D}, \text{E}\}$ l'ensemble des cinq joueurs.

   On note $\text{V}(x\,;\,y)$ le prédicat : « le joueur x a déjà battu le joueur y ».

   Ainsi, la valeur $\text{V}(\text{A}\,;\,\text{B})$ est VRAI, et la valeur de $\text{V}(\text{B}\,;\,\text{A})$ est FAUX.

   On définit trois prédicats :

   P1 : $\forall x \in \text{J},\ \exists y \in \text{J},\ x \neq y$ et $\text{V}(x\,;\,y)$

   P2 : $\exists x \in \text{J},\ \forall y \in \text{J},\ x \neq y$ et $\text{V}(x\,;\,y)$

   P3 : $\exists y \in \text{J},\ \forall x \in \text{J},\ x \neq y$ et $\text{V}(x\,;\,y)$

   Associer à chaque prédicat P1, P2, P3, celle des trois phrases suivantes qui lui correspond parmi les phrases suivantes. Aucune justification n'est demandée.

   « Il existe un joueur qui a été battu par tous les autres joueurs ».

   « Tous les joueurs ont battu au moins un autre joueur ».

   « Il existe un joueur qui a battu tous les autres joueurs ».

2. Un joueur reçoit un bonus lorsqu'il vérifie l'un au moins des trois critères suivants :
   - le joueur a participé à 20 parties ou davantage, et il a affronté plusieurs adversaires différents ;
   - le joueur n'a pas affronté plusieurs adversaires différents, et il a obtenu strictement plus de victoires que de défaites ;

- le joueur n'a pas obtenu strictement plus de victoires que de défaites, et il a participé à 20 parties ou davantage.

On définit les variables booléennes a, b, c de la façon suivante :

$a = 1$ si le joueur a participé à 20 parties ou davantage ; $a = 0$ sinon ;

$b = 1$ si le joueur a affronté plusieurs adversaires différents ; $b = 0$ sinon ;

$c = 1$ si le joueur a obtenu strictement plus de victoires que de défaites ; $c = 0$ sinon.

a. Écrire une expression booléenne F traduisant les conditions permettant à un joueur d'obtenir le bonus.

b. À l'aide d'un tableau de Karnaugh ou d'un calcul booléen, déterminer une écriture simplifiée de F sous forme d'une somme de deux termes.

c. En déduire une formulation simplifiée des critères permettant à un joueur d'obtenir le bonus.

Exercice 96

Soient a, b et c trois éléments d'une algèbre booléenne $\mathcal{B}$.

1. Soient $\mathrm{A} = ab + \overline{c}$ et $\mathrm{B} = \overline{a} + bc$, montrer par le calcul que

$$\overline{\mathrm{A}} = \overline{a}\,c + \overline{b}\,c$$

et que

$$\overline{\mathrm{B}} = a\overline{c} + a\overline{b}$$

2. Soit $\mathrm{C} = \overline{a}\,\overline{b}\,\overline{c} + a\overline{b}\,c + \overline{a}\,\overline{b} + a\overline{b}\,\overline{c}$.
Utiliser un diagramme de Karnaugh pour simplifier C.

Exercice 97

Une salle dédiée à l'informatique va être aménagée au lycée.
Le réseau qui équipera cette salle doit satisfaire au moins l'une des conditions suivantes :

- le réseau compte 5 postes ou plus et il existe un poste qui ne reçoit pas de données en entrée
- il existe un poste qui ne reçoit pas de données en entrée, et le réseau compte strictement moins de 5 postes, et il comporte strictement plus de 12 connexions ;
- le réseau comporte 12 connexions ou moins.

On définit les variables booléennes suivantes :

- $a = 1$ si le réseau compte 5 postes ou plus, $a = 0$ sinon ;
- $b = 1$ s'il existe un poste qui ne reçoit pas de données en entrée, $b = 0$ sinon ;
- $c = 1$ si le réseau comporte 12 connexions ou moins, $c = 0$ sinon.

1. Cette question est une question à choix multiple. Une seule réponse est correcte. Recopier sur la copie seulement la réponse correcte. On ne demande pas de justification.

Parmi les quatre phrases suivantes, donner celle qui traduit la variable $\overline{b}$:

- réponse A : « il existe un poste qui reçoit des données en entrée » ;

- réponse B : « tout poste reçoit des données en entrée » ;
- réponse C : « il existe un poste qui envoie des données en sortie » ;
- réponse D : « aucun poste ne reçoit des données en entrée ».

2. Donner l'expression booléenne E traduisant les critères voulus pour un réseau informatique.
3. À l'aide d'un tableau de Karnaugh ou par des calculs, exprimer E comme somme de deux variables booléennes.
4. Traduire les critères de sélection simplifiés, à partir de l'expression obtenue à la question 3.
5. Un réseau dans lequel 2 postes ne reçoivent pas de données en entrée et qui comporte 15 connexions répond-il aux critères voulus ? Justifier la réponse.

Exercice 98 : multiplexeur

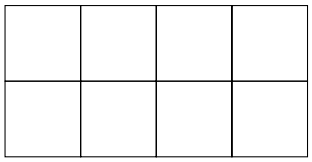
Un multiplexeur est un peu comme une télécommande à 2 boutons : on peut le résumer par une expression booléenne E dépendant de 3 variables a, b et c.

- Lorsque $a = 0$, la valeur de E est celle de b.
- Lorsque $b = 0$, la valeur de E est celle de c.

1. À l'aide de cette information, complète la table de vérité suivante.

a	b	c	E
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

2. Renseigne ensuite la table de Karnaugh



3. Déduis-en une expression simple de E.
4. Dessine le circuit électronique qui représente E.

Chapitre 7

Théorie des ensembles

1 Notions de base

Définition : ensemble, éléments

Nous nous contenterons de dire qu'un *ensemble* est une *collection d'objets* appelés *éléments* et qu'on peut clairement dire, lorsqu'on considère un objet, s'il appartient ou non à cet ensemble. Pour signifier que l'élément x appartient à l'ensemble E on écrit $x \in \mathrm{E}$. Pour signifier le contraire on écrit $x \notin \mathrm{E}$.

Exemples

- L'ensemble $\mathcal{J}$ des jours de la semaine peut se noter :

$$\mathcal{J} = \{ \text{ LUN } ; \text{ MAR } ; \text{ MER } ; \text{ JEU } ; \text{ VEN } ; \text{ SAM } ; \text{ DIM } \}$$

- On note $\mathbf{N}$ l'ensemble de tous les entiers positifs. On a donc $2020 \in \mathbf{N}$, mais $3,2 \notin \mathbf{N}$.
- Les solutions dans $\mathbf{R}$ de l'inéquation $x^2 < 4$ forment un ensemble : $]-2\ ;\ 2[$.

Définitions : ensemble vide et singleton

- L'ensemble qui ne contient aucun élément s'appelle *l'ensemble vide* et se note $\varnothing$.
- Soit x un objet, alors l'ensemble composé de l'unique élément x se note $\{x\}$ et s'appelle un *singleton*.

Définitions : écritures en extension et compréhension

Quand un ensemble est *fini* (on reviendra sur cette notion plus tard), on peut donner la liste *exhaustive* de ses éléments comme ceci :

E = {Paris ; Marseille ; Lyon ; Toulouse ; Nice ; Nantes ; Montpellier, Strasbourg, Bordeaux}
L'écriture précédente est appelée *écriture en extension*.

« E est l'ensemble des 9 plus grandes villes de France. »
Lorsqu'on écrit la phrase précédente, on donne une autre manière de définir E : pas par son contenu explicite, mais par une règle que ses éléments vérifient. On parle d'*écriture en compréhension*.

Exemples

- $\mathrm{E} = \{0;\ 1;\ 2\}$ est écrit en extension. En compréhension on écrira $\mathrm{E} = \{n \in \mathbf{N},\ n \leq 2\}$.
- On considère $\mathrm{F} = \{n \in \mathbf{N}, n \equiv 2,\ [3]\}$, c'est un ensemble *infini*, on ne peut donc pas écrire la

liste de tous ses éléments, mais on peut en donner un « aperçu » :

$$F = \{2; 5; 8; 11; 14; \dots\}$$

Définition

Quand un ensemble E est *fini*, on appelle *cardinal* de E le nombre de ses éléments, on le note Card(E).

Remarque

Un ensemble n'est pas une variable de type `list` en PYTHON, ou un tableau de VISUAL BASIC, car il n'y a *a priori* pas d'ordre sur les éléments de l'ensemble.
Ainsi l'ensemble { crayon ; stylo } est le même que l'ensemble { stylo ; crayon }.
En revanche en PYTHON, on a `['stylo', 'crayon']` et `['crayon', 'stylo']` sont deux listes différentes.

Définition : inclusion

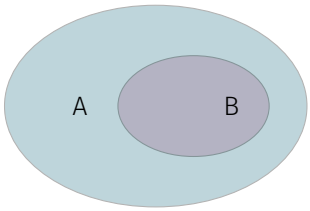
Soit A et B deux ensembles. Si tout élément de A est également un élément de B alors on dit que A *est inclus dans* B.
On écrit alors $A \subset B$ et cela revient à écrire : $\forall x \in A, x \in B$.

Exemples

- En reprenant un exemple précédent on a :

$$\{ \text{LUN} ; \text{JEU} ; \text{VEN} \} \subset \mathcal{J}$$

- Notons A l'ensemble des élèves de la classe et B l'ensemble des élèves de la classe qui habitent Saint-Brieuc.



$B \subset A$

Remarques

- Quel que soit l'ensemble E, on a $E \subset E$ car la proposition $\forall x \in E, x \in E$ est vraie.
- Quel que soit l'ensemble A, $\varnothing \subset A$: en effet la proposition suivante est vraie : $\forall x \in \varnothing, x \in A$. Cela peut paraître abusif étant donné qu'il n'y a aucun élément dans $\varnothing$. Faisons alors appel à nos cours de logique et montrons que la proposition contraire est fausse, ce qui revient au même. Cette proposition contraire s'énonce :
  $\exists x \in \varnothing, x \notin A$. Elle est évidemment fausse puisqu'on ne peut trouver $x \in \varnothing$.
- Il n'existe pas d'*ensemble de tous les ensembles* : on ne peut pas se dire que les ensembles peuvent tous être rangés dans un « gros ensemble » . En revanche (et c'est ce qu'on va faire), il est possible de se donner un ensemble de départ E et de considérer tous les ensembles inclus

dedans.

2 L'algèbre de Boole des parties d'un ensemble

Définition : ensemble des parties d'un ensemble

On se donne un ensemble E. L'ensemble de *tous les ensembles inclus dans* E se note $\mathcal{P}(E)$. On l'appelle aussi *ensemble des parties de* E.

- On a vu que $\varnothing \subset E$, ce qui se réécrit $\varnothing \in \mathcal{P}(E)$.
- De même $E \subset E$, ce qui se réécrit $E \in \mathcal{P}(E)$.

Exemple

Posons $E = \{a; b; c\}$ et donnons tous les éléments de $\mathcal{P}(E)$:

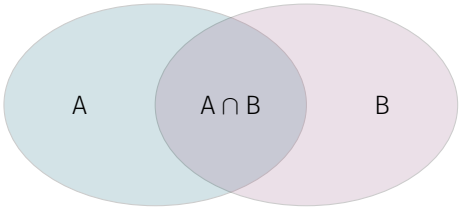
- il y a $\varnothing$;
- il y a $\{a\}$, $\{b\}$ et $\{c\}$;
- il y a $\{a; b\}$, $\{a; c\}$ et $\{b; c\}$, parties à 2 éléments.
- il y a E lui même.

Ainsi $\mathcal{P}(E) = \{\varnothing; \{a\}; \{b\}; \{c\}; \{a; b\}; \{a; c\}; \{b; c\}; \{a; b; c\}\}$ comporte 8 éléments.

Exercice 99

Déterminer $\mathcal{P}(E)$ lorsque $E = \{r; s; t; u; v\}$.

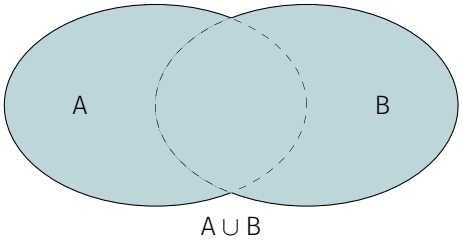
Définition : intersection



Soient A et B deux ensembles. On appelle *intersection* de A et de B et on note $A \cap B$ l'ensemble dont les éléments appartiennent à la fois à A et à B.

Définition : union

Soient A et B deux ensembles. On appelle *union* de A et de B et on note $A \cup B$ l'ensemble dont les éléments à A ou bien à B.



$A \cup B$

Exemple

Considérons les ensembles $A = \{ u ; v ; w \}$ et $B = \{ u ; w ; x ; y \}$.
Alors

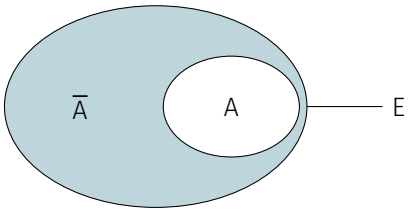
$$A \cap B = \{ u ; w \}$$

et

$$A \cup B = \{ u ; v ; w ; x ; y \}$$

Définition : complémentaire

Soit A une partie de E.
On appelle complémentaire de A dans E et on note $\overline{A}$ l'ensemble des éléments de E qui n'appartiennent pas à A.



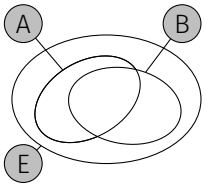
Exemple

En prenant $E = \{ \text{LUN} ; \text{MAR} ; \text{MER} ; \text{JEU} ; \text{VEN} ; \text{SAM} ; \text{DIM} \}$ comme ensemble de départ, le complémentaire dans E de {LUN; JEU; VEN} est {MAR; MER; SAM; DIM}.

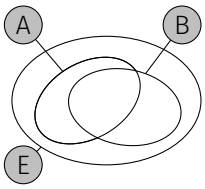
Exercice 100

A et B sont deux parties de E. Colorier l'ensemble demandé.

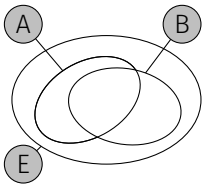
1. $A \cup \overline{B}$



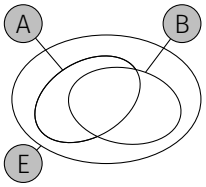
2. $A \cap \overline{B}$



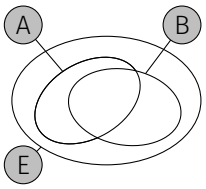
3. $\overline{A \cap B}$



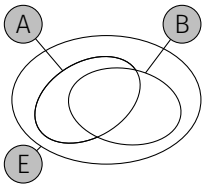
4. $\overline{A \cup B}$



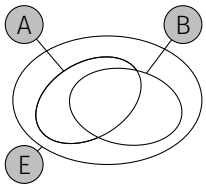
5. $\overline{A} \cap \overline{B}$



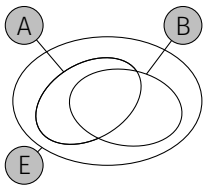
6. $\overline{A} \cup \overline{B}$



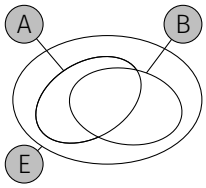
7. $\overline{A \cap \overline{B}}$



8. $(A \cap \overline{B}) \cup (A \cap B)$



9. $(A \cap \overline{B}) \cup (\overline{A} \cap B)$



Remarque

Quand on considère 2 parties d'un ensemble E, leur union en est encore une, de même que leur intersection. Il en va encore de même pour le complémentaire d'une partie de E. Ces trois opérations sont donc « internes » à $\mathcal{P}(E)$, et on peut même énoncer le résultat suivant :

Propriété

Soit E un ensemble. Alors lorsqu'on munit $\mathcal{P}(E)$ de $\cup$, $\cap$ et de l'opération « complémentaire » , on obtient une *algèbre de Boole* :

- 0 se note $\varnothing$;
- 1 se note E ;
- + se note $\cup$;
- $\times$ se note $\cap$.

Cela implique qu'on peut effectuer les calculs dans $\mathcal{P}(E)$ de la même manière qu'on effectue les calculs booléens.

Exemple

$$\begin{aligned} A \cup (B \cap \overline{A}) &= (A \cup B) \cap (A \cup \overline{A}) && \text{car } \cup \text{ se distribue sur } \cap \\ &= (A \cup B) \cap E && \text{puisque } A \cup \overline{A} = E \text{ (tout comme } a + \overline{a} = 1) \\ &= A \cup B && \text{car E est neutre pour } \cap \\ & && \text{(tout comme 1 est neutre pour } \times) \end{aligned}$$

On peut aussi utiliser des *diagrammes de Venn* (les dessins « à base de patatoïdes » vus plus haut) dans les cas simples.

Exercice 101

Illustrer à l'aide d'un diagramme que l'égalité $A \cap B = A \cap C$ n'entraîne pas automatiquement $B = C$.

Propriété : lois de De Morgan

Pour toutes parties A et B de E on a

$$\overline{A \cup B} = \overline{A} \cap \overline{B} \qquad \text{et} \qquad \overline{A \cap B} = \overline{A} \cup \overline{B}$$

puisque $\mathcal{P}(E)$ est une algèbre de Boole.

Méthode

Soient A et B deux parties de E, comment écrire l'ensemble

$$F = \{x \in E,\ x \in A \text{ et } x \notin B \cup C\}$$

à l'aide d'union, intersection et complémentaire ?

- les « et » représentent des $\cap$;
- les « ou » représentent de $\cup$;
- pour tout ensemble G, $x \notin G$ équivaut à $x \in \overline{G}$.

Donc dans ce cas

$F = A \cap \overline{B \cup C}$ et on utilise une loi de De Morgan
$= A \cap \overline{B} \cap \overline{C}$

Exercice 102

Dans $\mathcal{P}(E)$ on définit l'opération Δ comme ceci :

$$A \Delta B = \{x \in E,\ x \in A \cup B \text{ et } x \notin A \cap B\}$$

1. Faire un diagramme de Venn et colorier $A \Delta B$.
2. À quelle opération logique Δ correspond-elle ?
3. Exprimer Δ à l'aide de $\cup$, $\cap$ et le complémentaire.
4. Montrer par le calcul que $(A \Delta B) \Delta C = A \Delta (B \Delta C)$.

Exercice 103

Dans $\mathcal{P}(E)$ on définit l'opération $\setminus$ comme ceci :

$$A \setminus B = \{x \in E,\ x \in A \text{ et } x \notin B\}$$

1. Faire un diagramme de Venn et colorier $A \setminus B$.
2. Exprimer $\setminus$ à l'aide de $\cup$, $\cap$ et le complémentaire.
3. Montrer par le calcul que $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$.

3 Produit cartésien

Définition : produit cartésien de deux ensembles

Soient E et F deux ensembles, on appelle *produit cartésien* de E par F et on note $E \times F$ l'ensemble des couples $(x; y)$, où $x \in E$ et $y \in F$.

$$E \times F = \{(x; y),\ x \in E,\ y \in F\}$$

Exemples

- Prenons $E = \{bille; ballon\}$ et $F = \{rouge; vert; bleu\}$ alors on peut représenter $E \times F$ de la manière suivante :

	bille	**ballon**
rouge	(bille ; rouge)	(ballon ; rouge)
vert	(bille ; vert)	(ballon ; vert)
bleu	(bille ; bleu)	(ballon ; bleu)

- $\mathbf{R} \times \mathbf{R}$ se note aussi $\mathbf{R}^2$. Quand on s'est donné un repère du plan $(O ; I ; J)$, alors tout point du plan possède un unique couple de coordonnées dans $\mathbf{R}^2$ et réciproquement, à tout couple de $\mathbf{R}^2$ correspond un unique point du plan. C'est pourquoi on identifie $\mathbf{R}^2$ au plan... et $\mathbf{R}^3$ à l'espace.
- Le produit cartésien est fondamental dans le domaine des bases de données.

Exercice 104

En prenant $E = \{1; 2\}$ et $F = \{a; b; c\}$ construire $E \times F$ et $F \times E$ et montrer que ces ensembles sont différents.

Propriété

Si $\text{Card}(E) = n$ et $\text{Card}(F) = p$ alors $\text{Card}(E \times F) = n \times p$.

Définition : produit cartésien de plusieurs ensembles

Soient $E_1, ...E_n$ k ensembles (n entier supérieur à 2), on note

$$\prod_{k=1}^{k=n} E_k = E_1 \times ... \times E_n$$

l'ensemble des n-*uplets* $(x_1; ... ; x_n)$ ou chaque x_k appartient à E_k.
On dit que c'est le *produit cartésien* des ensembles E_k.

Propriété

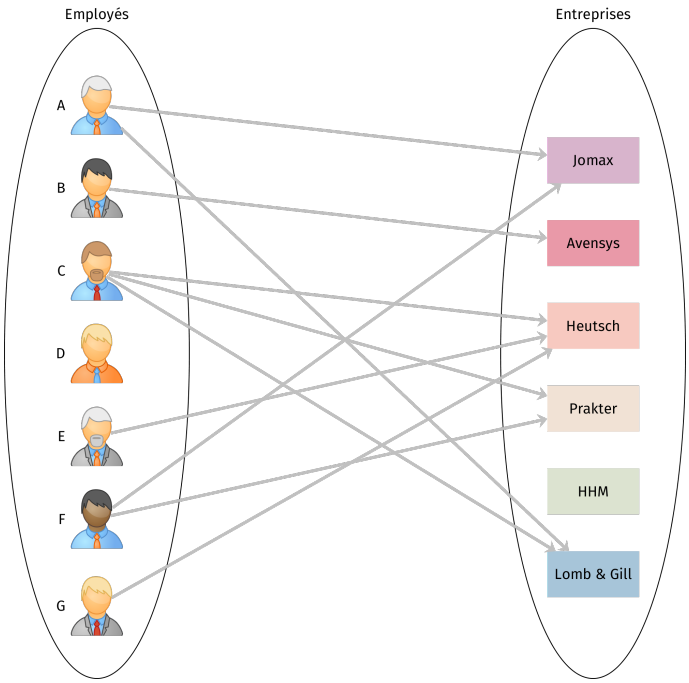
Quand tous les ensembles sont finis, le cardinal de l'ensemble produit s'obtient en multipliant les cardinaux des ensembles.

4 Relations binaires

Définition : relation binaire

Soient E et F deux ensembles. Une *relation binaire de* E *vers* F c'est la donnée de certains *couples* $(x_i; y_i)$; où $x_i \in$ E et $y_i \in$ F.
E est appelé *ensemble de départ* et F *ensemble d'arrivée*. L'ensemble de ces couples (notons-le G), est appelé le *graphe* de la relation. C'est une partie de E $\times$ F.

Exemple : avec un diagramme sagittal



On considère un ensemble E d'employés, un ensemble F d'entreprises et la relation « … est ou a été un employé de … » .

- (A; *Jomax*) $\in$ G : l'employé A a travaillé pour l'entreprise Jomax ;
- on a aussi (A; *Lomb & Gill*) $\in$ G car cet employé a aussi occupé un poste là-bas ;
- D n'a pas occupé de postes dans F.
- HHM n'a aucun employé de E.

G est ici représenté par l'ensemble des flèches.

Exemple : avec un tableau

Voici la même relation :

	A	B	C	D	E	F	G
Jomax	x					x	
Avensys		x					
Heutsch			x		x		x
Prakter			x			x	
HHM							
Lomb & Gill	x		x				

Nous étudierons plus en détail les relations binaires pour lesquelles l'ensemble de départ et d'arrivée sont les mêmes.

Définition : relation binaire dans un ensemble

Quand $E = F$ on parle de *relation binaire dans* E.

Exemples

- L'égalité de deux entiers naturels est une relation binaire dans $\mathbf{N}$.
- Posons $E = \mathbf{N}^*$ et disons que $x\mathcal{R}y$ si et seulement si x divise y. On obtient une relation binaire.
- Dans l'ensemble $\mathcal{D}$ des droites du plan disons que $d\mathcal{R}d'$ si et seulement si $d \perp d'$. On obtient une relation binaire.

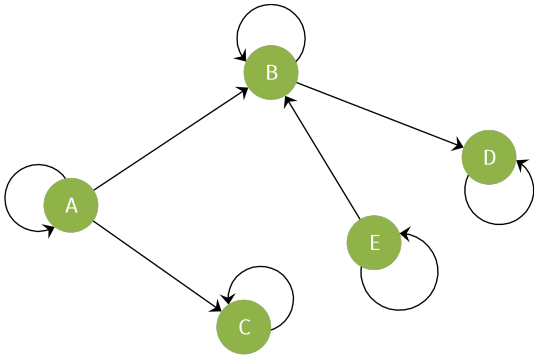
Définitions : réflexivité, symétrie, antisymétrie, transitivité

Soit $\mathcal{R}$ une relation binaire sur E, on dit que $\mathcal{R}$ est

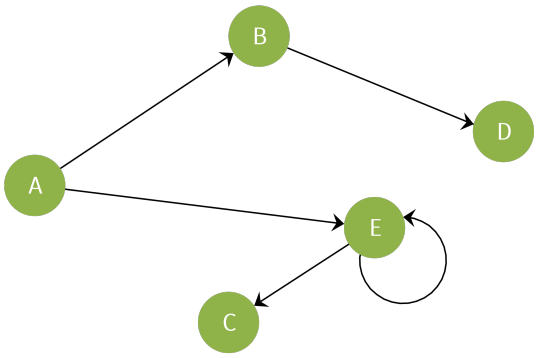
- *réflexive* si tout élément est en relation avec lui-même : $\forall x \in E$, $x\mathcal{R}x$;
- *symétrique* si dès que $x\mathcal{R}y$, cela entraîne $y\mathcal{R}x$ (et vice versa, évidemment) ;
- *antisymétrique* si deux éléments différents ne peuvent être en relation « dans les deux sens » . Cela revient à dire que si on trouve $x\mathcal{R}y$ et $y\mathcal{R}x$, alors nécessairement $x = y$;
- *transitive* si lorsque $x\mathcal{R}y$ et $y\mathcal{R}z$ alors on a $x\mathcal{R}z$.

Exemples

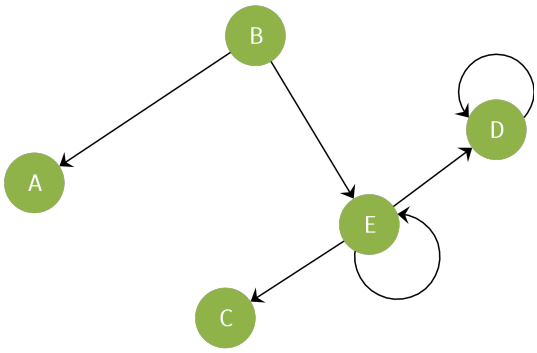
$\mathcal{R}$ est réflexive. Tout élément est en relation avec lui même :



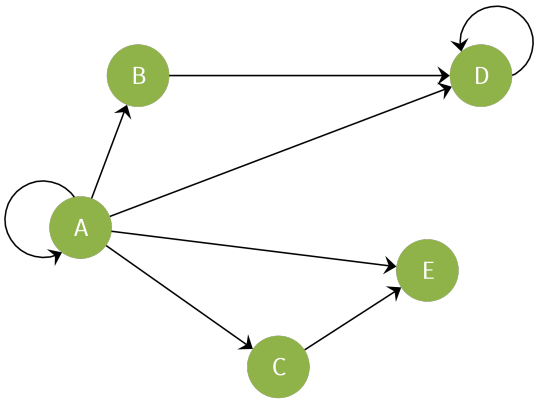
$\mathcal{R}$ est symétrique. Si $x\mathcal{R}y$ alors $y\mathcal{R}x$ de sorte que les éléments en relation le sont « dans les deux sens » :



$\mathcal{R}$ est antisymétrique. Si une relation est vraie « dans les deux sens » alors c'est qu'elle ne concerne qu'un élément : il n'y a pas de double flèche reliant deux éléments différents mais, peut-être, des cas comme D$\mathcal{R}$D ou E$\mathcal{R}$E :



$\mathcal{R}$ est transitive. À chaque fois qu'on peut « enchaîner » les relations, le « raccourci » est présent aussi :

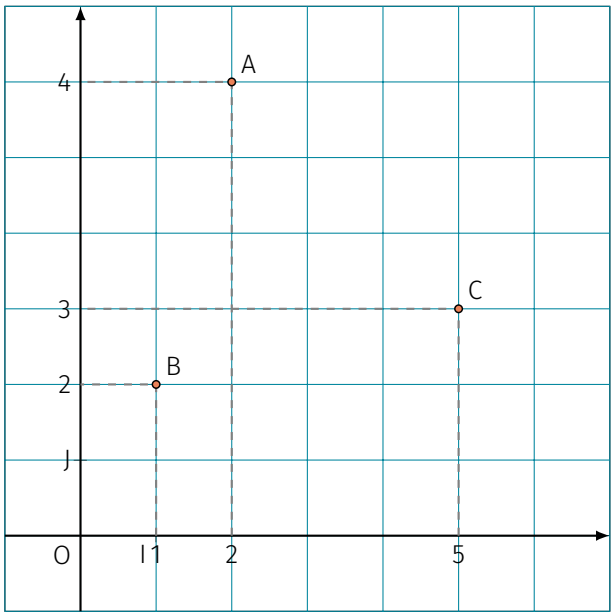


Définitions : relation d'équivalence, relation d'ordre

- Lorsqu'une relation binaire dans E est réflexive, symétrique et transitive, on dit que c'est une relation d'équivalence sur E.
  Une relation d'équivalence permet de partager E en *classes* d'éléments qui sont tous équivalents (considérés « pareils » pour la relation).
- Lorsqu'une relation binaire dans E est réflexive, antisymétrique et transitive, on dit que c'est une relation d'ordre sur E.
  Une relation d'ordre permet de « ranger les éléments comparables » :
  - Si à chaque fois qu'on prend deux éléments *distincts* x et y alors on a $x\mathcal{R}y$ ou $y\mathcal{R}x$ alors l'ordre est dit *total* car on peut toujours comparer deux éléments différents.
  - Si ce n'est pas le cas on parle d'ordre partiel.

Exemples

- Dans N, la relation d'égalité est une relation d'équivalence.
- Dans **R** la relation $\leqslant$ est une relation d'ordre totale.
- Dans l'ensemble des droites du plan, la relation de parallélisme est une relation d'équivalence.
- On considère $\mathbf{R}^2$ et on l'identifie à l'ensemble des points du plan muni d'un repère $(\,O\,;\,I\,;\,J\,)$. On décide de noter $\preceq$ la relation suivante :
  $(x_1, y_1) \preceq (x_2, y_2)$ si et seulement si $x_1 \leqslant x_2$ et $y_1 \leqslant y_2$.



alors $\preceq$ est une relation d'ordre, mais c'est ordre n'est pas total : On a bien $(1\,;\,2) \preceq (2\,;\,4)$ mais on ne peut pas comparer $(2\,;\,4)$ et $(5\,;\,3)$.

Question

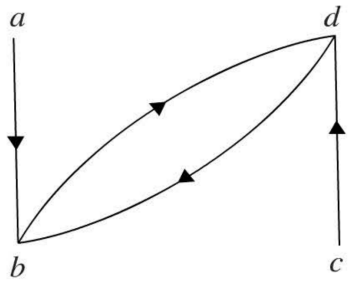
Dans **R**, la relation < est-elle une relation d'ordre ?

Exercice 105

La relation binaire $\mathcal{R}$ est elle réflexive, symétrique, antisymétrique, transitive ? Est-ce une relation d'équivalence ou d'ordre ? Si c'est une relation d'ordre est elle totale ou partielle ?

- Dans **R**, $x\mathcal{R}y \Leftrightarrow x < y$.
- Dans **N**, $x\mathcal{R}y \Leftrightarrow x$ divise y.
- Soit E un ensemble, dans $\mathcal{P}(\mathrm{E})$, $\mathrm{A}\mathcal{R}\mathrm{B} \Leftrightarrow \mathrm{A} \subset \mathrm{B}$.
- Soit E un ensemble, dans $\mathcal{P}(\mathrm{E})$, $\mathrm{A}\mathcal{R}\mathrm{B} \Leftrightarrow \mathrm{A} \cap \mathrm{B} = \varnothing$.

Exercice 106



On considère le dessin comme étant la représentation d'une relation binaire, que l'on note $\mathcal{R}$. définie sur l'ensemble $S = \{a; b; c; d\}$.

a. Écrire tous les éléments qui sont en relation sou s la forme $x\mathcal{R}y$, avec $(x; y) \in S^2$.

b. La relation R est elle réflexive ?

c. La relation R est elle symétrique ?

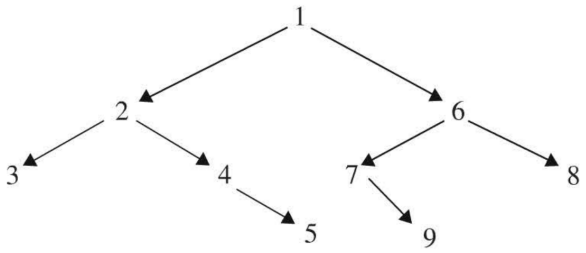
d. La relation R est elle transitive ?

e. Au minimum, quelles flèches doit-on ajouter pour obtenir la représentation d'une relation réflexive ?

f. Même question avec une relation symétrique.

Exercice 107

On considère l'*arbre binaire suivant* et sur l'ensemble des nombres présents dans l'arbre, on définit une relation binaire : $x\mathcal{R}y$ si et seulement si $x = y$ on bien on peut passer de x à y ou de y à x par un chemin qui descend toujours par la droite.



1. Expliquer pourquoi simplement à partir de sa définition on peut affirmer que $\mathcal{R}$ est réflexive et symétrique.
2. Montrer que $\mathcal{R}$ est une relation d'équivalence.
3. Regrouper sur le schéma ci-dessus les nombres équivalents.

Exercice 108

Dire si les relations suivantes sont réflexives, symétriques, antisymétriques, transitives.
Dire ensuite si ce sont des relations d'équivalence, d'ordre total ou partiel.

1. Sur $\mathbf{Z}$, $x\mathcal{R}y \iff x = -y$.
2. Sur $\mathbf{R}^2$, $(x, y)\mathcal{R}(x' y') \iff x = x'$.

3. Soit E un ensemble, sur $\mathcal{P}(E)$, $X\mathcal{R}Y \iff X = Y$ ou $X = \overline{Y}$.
4. Sur $\mathbb{Z}$, $x\mathcal{R}y \iff x + y$ est pair.
5. Soit E un ensemble et $A \subset E$, sur $\mathcal{P}(E)$, $X\mathcal{R}Y \iff X \cup A = Y \cup A$.

5 Applications

Définitions : application image, antécédents

Soient E et F deux ensembles. On appelle *application* de E dans F une relation $\mathcal{R}$ binaire de E vers F telle que pour tout élément x de E *il existe un unique* y de F tel que $x\mathcal{R}y$.

L'usage est alors de noter $\mathcal{R}$ comme *une fonction* :

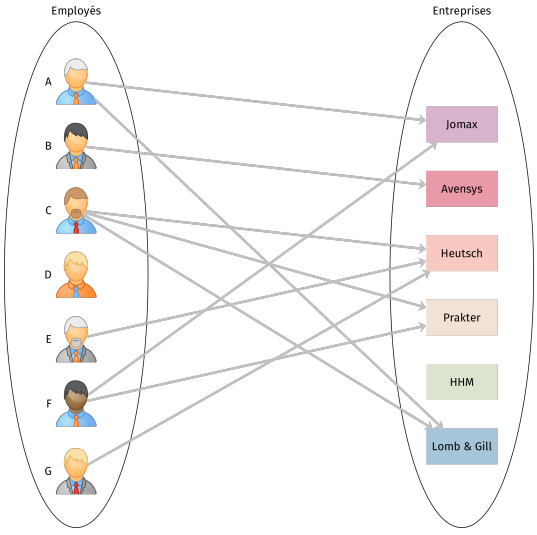
$f : E \longrightarrow F$
$x \longmapsto y$ où y est l'unique élément de F tel que $x\mathcal{R}y$.

Et on écrit que $y = f(x)$.

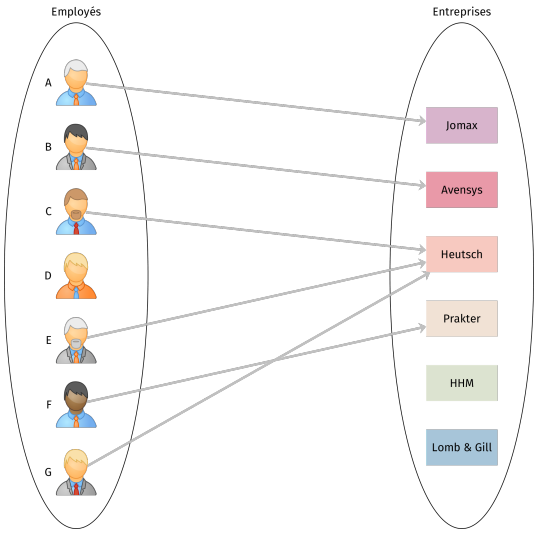
y est appelé l'*image* de x par l'application f.
On dit que x est *un antécédent* de y par f.

Exemples

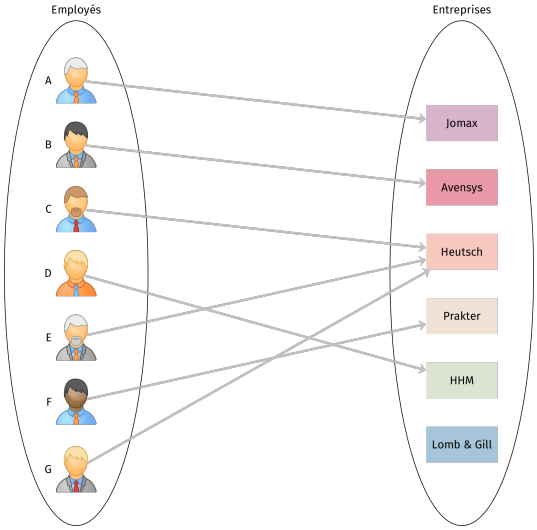
Cette relation binaire n'est pas une application de l'ensemble Employés dans l'ensemble Entreprises car (par exemple), l'élément A est associé à 2 éléments de Entreprises



Celle-ci n'en est pas une non plus car D n'est associé à aucun élément de Entreprises.



Celle-ci en est une car tout élément de Employés est associé à un *unique* élément de Entreprises.



Si on décide de l'appeler f alors on écrira :
$f(\mathrm{A}) = Jomax$ et de même $f(\mathrm{B}) = Avensys$.

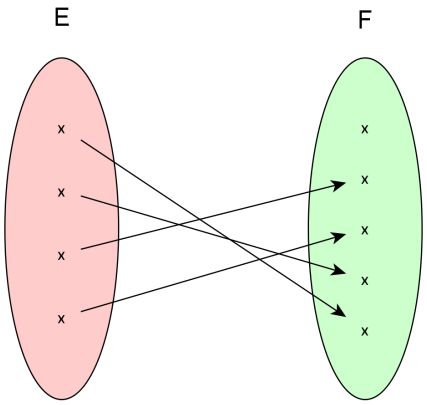
Remarque

Lorsque f est une application de E dans F, *tous* les éléments de E ont *une unique image* dans F. En revanche, tous les éléments de l'ensemble d'arrivée F n'ont pas obligatoirement un unique antécédent par f : chacun peut en avoir aucun, un seul ou plusieurs.
Dans l'exemple précédent Jomax admet A pour unique antécédent. Heutsch admet 3 antécédents, et Lomb & Gill n'en a aucun.

Définitions : injection, surjection, bijection

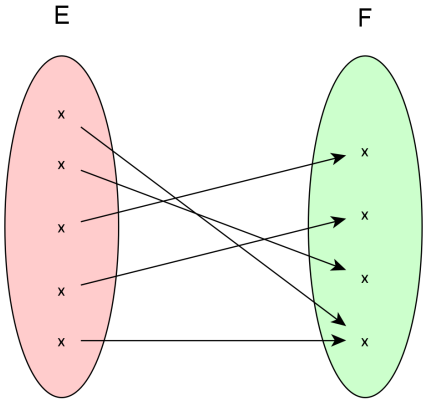
Soit f une application de E dans F.

- Si tout élément de F admet *au plus* un antécédent par f alors on dit que f est *injective* ou bien que c'est une *injection*.



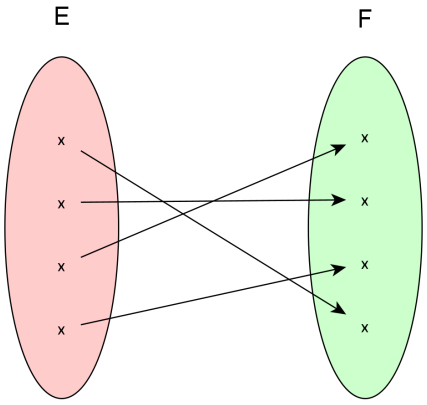
injection de E dans F

– Si tout élément de F admet *au moins* un antécédent par *f* alors on dit que *f* est *surjective* ou bien que c'est une *surjection*.



surjection de E dans F

– tout élément de F admet *exactement* un antécédent par *f* alors *f* est *à la fois injective et surjective* et on dit que *f* est *bijective* ou bien que c'est une *bijection*.

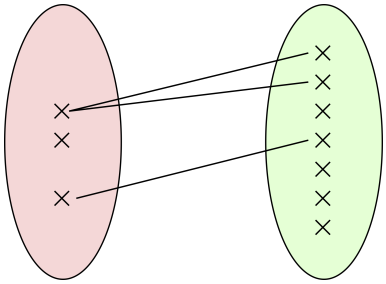


bijection de E dans F

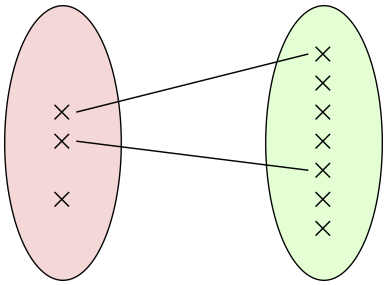
Exercice 109

Pour chaque relation de E (en rose à gauche) vers F (en vert à droite) indiquer si c'est une application, et si elle est injective, surjective, bijective ou rien du tout.

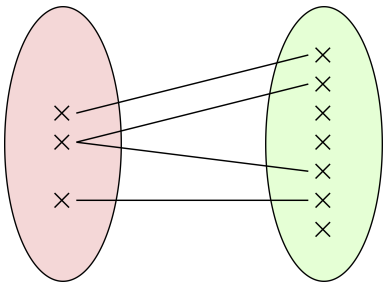
1.



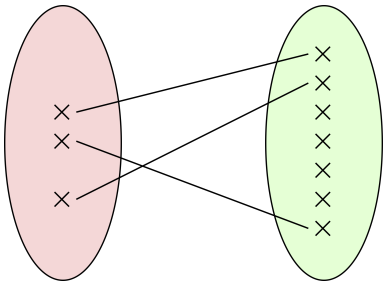
2.



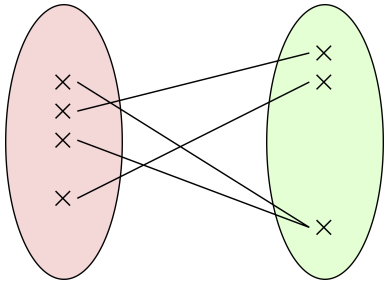
3.



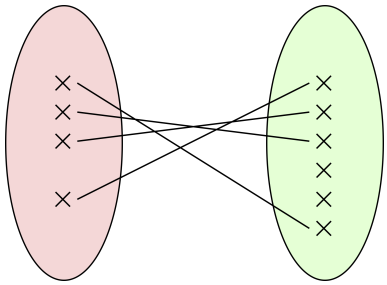
4.



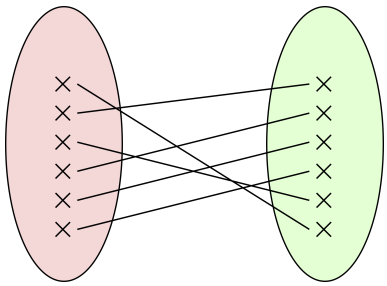
5.



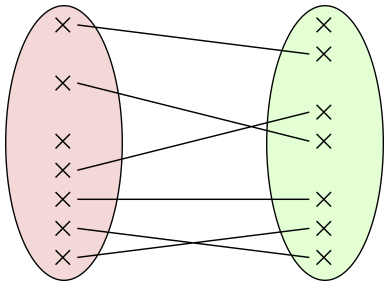
6.



7.



8.



Propriété : bijection réciproque

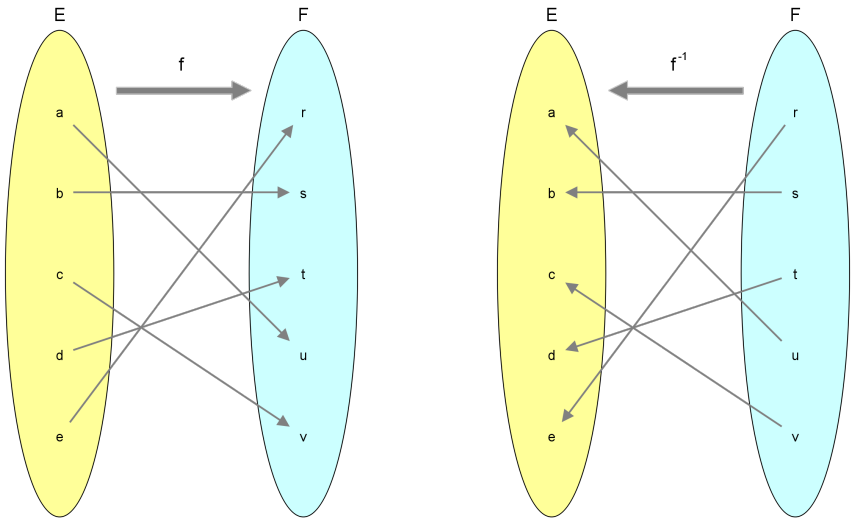
Lorsque f est une bijection de E dans F, il est possible de construire sa *bijection réciproque*. On la note f^{-1}, elle part de F, arrive dans E et à tout élément y de F elle associe *son unique antécédent* par f.

$f^{-1} : \mathrm{F} \longrightarrow \mathrm{E}$
$y \longmapsto x$ où x est l'unique élément de E tel que $f(x) = y$.

Et on écrit que $x = f^{-1}(y)$.

Exemple

f est une bijection de E dans F : chaque élément de F possède *un unique* antécédent par f.
Cela permet de construire la *bijection réciproque* de f, notée f^{-1}. Celle-ci va de F vers E. Puisque $f(e) = r$, on pose alors $f^{-1}(r) = e$, *et cætera*.



Il va sans dire que f^{-1} est également une bijection.

6 Extension aux parties d'une application

Définition : image directe

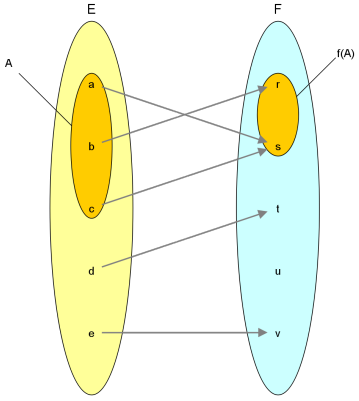
Soit f une application de E dans F et A une partie de E. On appelle *image directe de* A *par* f et on note $f(\mathrm{A})$ la partie de F constituée des images des éléments de A par f.

$$f(\mathrm{A}) = \{f(x) \ : \ x \in \mathrm{A}\}$$

Exemple

A est la partie de E constituée de a, b, c.
On considère B, partie de F constituée de $f(a) = s$, $f(b) = r$ et $f(c) = s$ (élément déjà atteint par a).

$$\mathrm{B} = f(\mathrm{A})$$

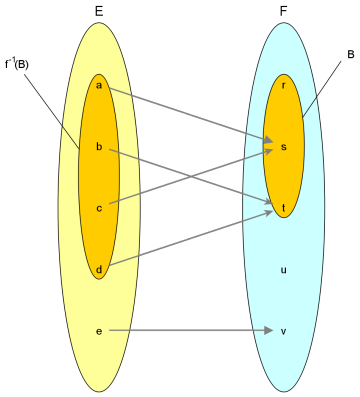


Définition : image réciproque

Soit f une application de E dans F et B une partie de F. On appelle *image réciproque de* B *par* f et on note $f^{-1}(B)$ la partie de E constituée des *antécédents* des éléments de B par f.

$$f^{-1}(B) = \{x \in E \ : \ f(x) \in B\}$$

Exemple



B est la partie de F constituée de r, s, t.
r n'a pas d'antécédent par f, s en a deux : a et c, et t en a également 2 : b et d.

$$f^{-1}(B) = \{a; \ b; \ c; \ d\}$$

Remarque

La notation f^{-1} est trompeuse : f n'est pas obligatoirement bijective, donc l'*application* f^{-1} n'est pas obligatoirement définie. Ceci dit il est possible de déterminer $f^{-1}(B)$ quand B est une *partie* de F même si f n'est pas bijective.

Exercice 110

$E = \{0; 1; 2; 3; 4; 5; 6; 7\}$ et $F = \{0; 1; 2; 3\}$.
f est l'application de E dans F qui à tout élément de E associe son reste dans la division euclidienne par 3.

1. f est-elle injective ? Surjective ?
2. Posons $A = \{1; 3; 4\}$, déterminer $f(A)$, puis posant $B = f(A)$, déterminer $f^{-1}(B)$.
3. Posons $C = \{2; 3\}$, déterminer $f^{-1}(C)$, puis, en posant $D = f^{-1}(C)$, déterminer $f(D)$.

Exercice 111

Soit f l'application de $\mathbf{R}$. dans $\mathbf{R}$. définie par $f(x) = 4x + 10$.

1. f est-elle une injection ?
2. f est-elle une surjection ?
3. f est-elle une bijection ?
4. Déterminer l'image directe de $[2 \ ; \ 3]$ et de $[0 \ ; \ +\infty[$.

5. Déterminer l'image réciproque de $[0\ ;\ +\infty[$.

7 Composition

Définition : composée de deux applications

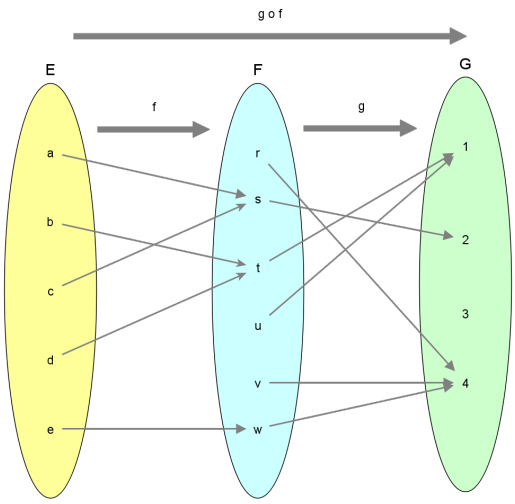
Soit f une application de E dans F et g une application de F dans G.
Alors on définit la *composée* de g par f et on note $g \circ f$ l'application de E dans G définie par

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

Exemple

Les applications f et g sont décrites par le diagramme ci-contre.
L'application $g \circ f$ est donc définie de E dans G et

- $g \circ f(a) = 2$
- $g \circ f(b) = 1$
- $g \circ f(c) = 2$
- $g \circ f(d) = 1$
- $g \circ f(e) = 4$



Exercice 112

E $= \{a; b; c; d\}$, F $= \{1; 2; 3\}$ et G$\{\alpha; \beta; \gamma\}$.
f est définie de E dans F et g de F dans G par
$f(a) = 2$, $f(b) = 1$, $f(c) = 3$ et $g(1) = \gamma$, $g(2) = \alpha$ et $g(3) = \beta$.

1. f et g sont-elles injectives ? Surjectives ? Bijectives ?
2. Définir l'application $g \circ f$.
3. Peut-on définir l'application réciproque de f ? De g ?

Exercice 113*

Expliciter $f \circ g$ et $g \circ f$ lorsque f et g sont les fonctions suivantes :

$$\begin{aligned} f \ : \ \mathbf{R} &\longrightarrow]0\ ;\ +\infty[\\ x &\longmapsto x^2+2 \end{aligned} \qquad \begin{aligned} g \ : \ ]0\ ;\ +\infty[&\longrightarrow \mathbf{R} \\ x &\longmapsto \frac{1}{\sqrt{x}-1} \end{aligned}$$

Exercice 114

Un administrateur réseau gère le parc d'une petite entreprise qui comprend 9 ordinateurs. Chaque ordinateur possède une adresse de carte réseau, dite adresse MAC (Media Access Control) unique. L'administrateur a assigné une adresse IP (Internet Protocol) à chaque ordinateur à l'aide d'un logiciel installé sur le serveur. Il obtient le tableau suivant :

Adresse MAC	n° de poste	Adresse IP
00 : FF : B4 : A9 : 96 : 11	1	172.16.0.21
00 : FF : B4 : B0 : 45 : 1A	2	172.16.0.22
00 : FF : B4 : 00 : C5 : DE	3	172.16.0.23
00 : EE : B5 : 01 : 32 : C4	4	172.16.0.24
00 : EE : B5 : 01 : 32 : C5	5	172.16.0.25
00 : EE : B5 : 01 : 32 : C6	6	172.16.0.26
00 : FF : B4 : 00 : C5 : DF	7	172.16.0.27
00 : FF : B4 : 00 : 02 : 98	8	172.16.0.28
00 : EE : B5 : 01 : 34 : CA	9	172.16.0.29

On considère l'application f qui, à un numéro d'ordinateur, associe la dernière partie de l'adresse IP.
Cette dernière partie est un entier variant de 2 à 255.

$$f : \{1\ ;\ 2\ ;\ 3\ ;\ 4\ ;\ 5\ ;\ 6\ ;\ 7\ ;\ 8\ ;\ 9\} \longrightarrow \{2\ ;\ 3\ ;\ \ldots\ ;\ 255\}.$$

Par exemple, $f(1) = 21$.

1. Justifier le fait que cette application est injective.
2. Cette application est-elle surjective ? Justifier.
3. À la suite d'une opération informatique, le poste dont l'adresse MAC est

   00 : FF : B4 : 00 : C5 : DF obtient l'adresse IP suivante : 172.16.0.23. Les autres postes gardent leur adresse IP précédente.

   On a alors une nouvelle application

   $$g : \{1\ ;\ 2\ ;\ 3\ ;\ 4\ ;\ 5\ ;\ 6\ ;\ 7\ ;\ 8\ ;\ 9\} \longrightarrow \{2\ ;\ 3\ ;\ \ldots\ ;\ 255\}.$$

   L'application g est-elle injective ? Justifier.

Chapitre 8

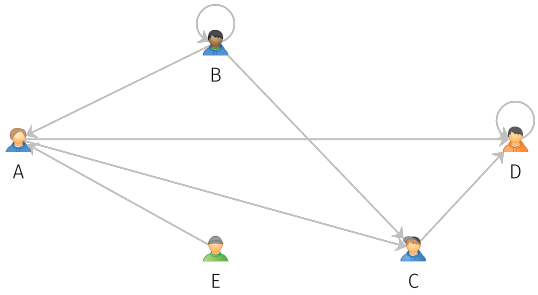
Graphes

1 Introduction

1.1 Plusieurs représentations

Abel, Brieuc, Corentin, David et Ewen postent des messages sur un réseau social. Un message peut-être « aimé » par n'importe quel utilisateur, y compris son créateur.
On regarde, sur une période de deux semaines, qui a aimé les messages de qui. Voici les résultats :

- Abel a aimé des messages de Corentin et David ;
- Brieuc a aimé ses messages et ceux de Corentin ;
- Corentin a aimé les messages de David ;
- David a aimé ses propres messages ;
- Ewen a aimé les messages d'Abel.



Ces résultats permettent de produire un *graphe orienté* :

- les *sommets du graphe* représentent les personnes ;
- les *arêtes* sont des flèches qui représentent le fait que la personne de départ aime les messages de celle d'arrivée.

On peut aussi représenter les résultats dans un tableau :

	A	B	C	D	E
aime les messages de	C,D	B,C	D	D	A

On peut aussi recopier le tableau en donnant pour chaque personne la liste de ses « followers » (personnes qui ont aimé ses messages) :

	A	B	C	D	E
followers	B,E	B	A,B	A,C,D	—

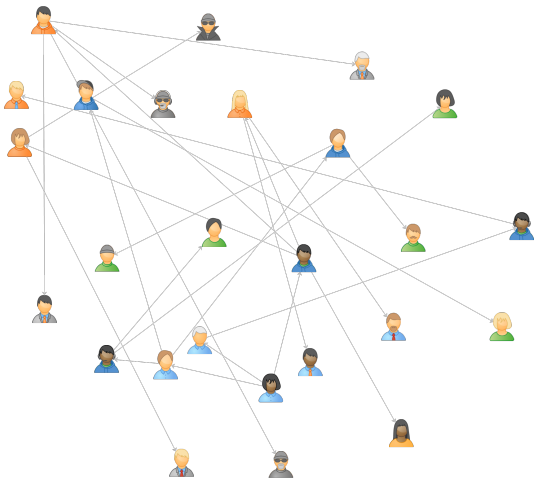
On peut aussi présenter les données ainsi :

		arrivée				
		A	B	C	D	E
départ	A	0	0	1	1	0
	B	0	1	1	0	0
	C	0	0	0	1	0
	D	0	0	0	1	0
	E	1	0	0	0	0

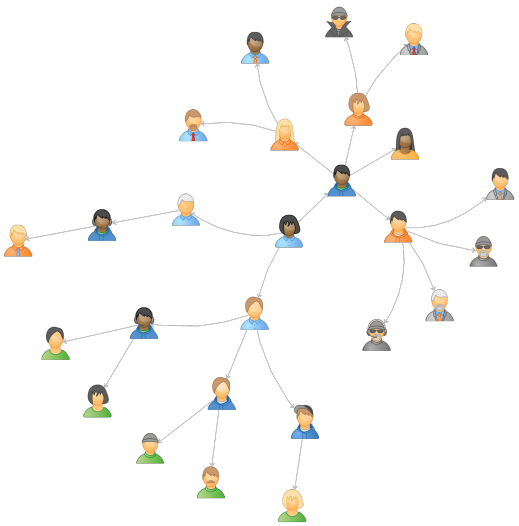
Il y a donc plusieurs manières de représenter un graphe orienté.

1.2 Organiser un graphe orienté

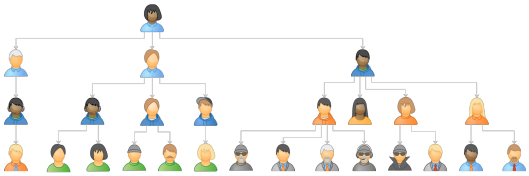
On considère des individus qui ont infectés par une maladie et on place une flèche pour signifier que tel individu a contaminé tel autre individu.
On obtient ce résultat :



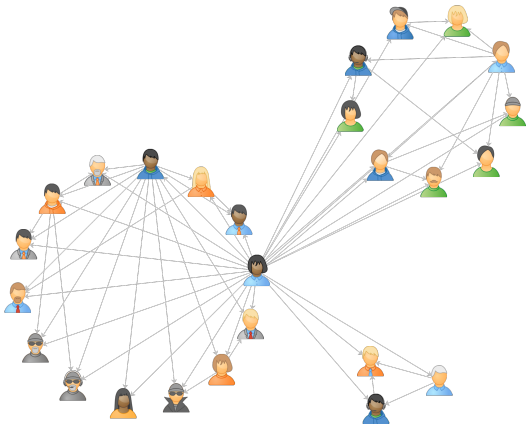
C'est le fouillis, n'est-ce pas ?
En déplaçant les sommets du graphe on peut le présenter ainsi :



C'est le même graphe mais on y voit plus clair... et on y voit encore plus clair comme ça :



On voit clairement le « patient zéro » . On peut aussi rajouter des flèches quand la contamination n'est pas directe mais s'est faite « par une ou plusieurs personnes interposées » . On obtient ceci



Le but de ce chapitre est de formaliser tout cela (et plus).

2 Représentations d'un graphe orienté

2.1 Premières notions

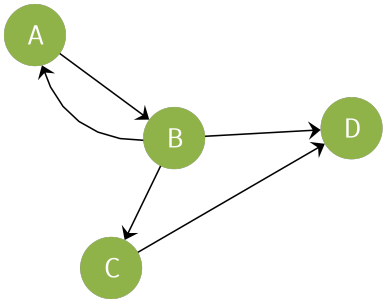
Définition : graphe, sommet, arc

Un graphe orienté, c'est la donnée de

- un ensemble $S = \{s_1; ...; s_n\}$ de *sommets* ;
- un ensemble A composé d'*arcs* du type $(s_i; s_j)$, qui indiquent qu'il y a « une flèche » partant du sommet s_i et allant au sommet s_j. :
  s_i est appelé l'*origine* de l'arc et s_j son *extrémité*.

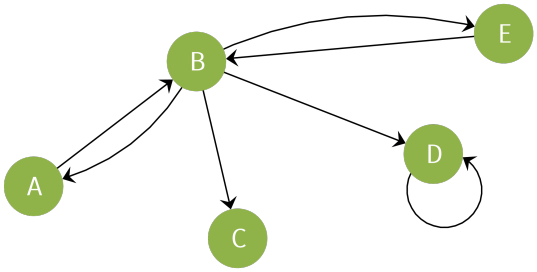
Exemple

Ici l'ensemble des sommets est $\{A; B; C; D\}$.
L'ensemble des arcs est :
$\{(A; B); (B; A); (B; C); (B; D); (C; D)\}$.



Exercice 115

Donner l'ensemble des sommets et l'ensemble des arêtes du graphe suivant.



Définition : boucle

Un arc dont l'origine et l'extrémité sont confondues d'appelle une *boucle*.

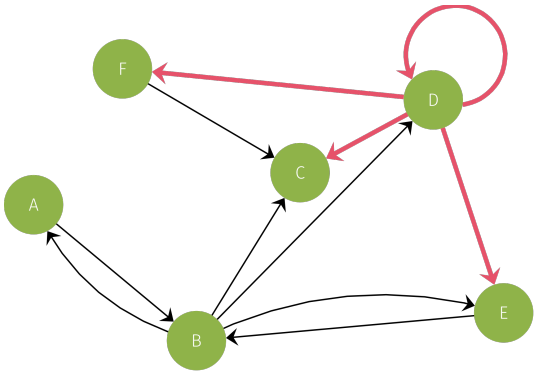
2.2 Successeurs, prédécesseurs

Définition : successeur, prédécesseur

Soit un graphe de sommets S et d'arcs A. Soit s un sommet.
On note $\Gamma^+(s)$ l'ensemble des *successeurs de* s.
C'est l'ensemble des extrémités des arcs *partant de* s.
De même, on note $\Gamma^-(s)$ l'ensemble des *prédécesseurs de* s.
C'est l'ensemble des origines des arcs *arrivant sur* s.

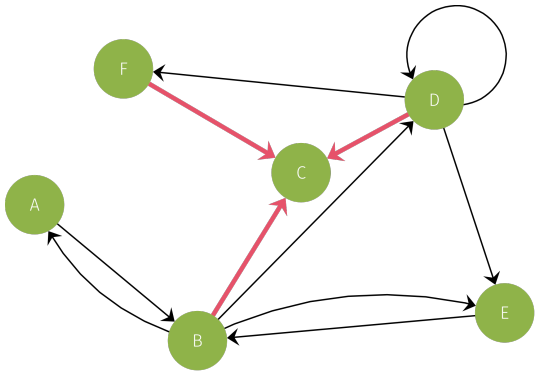
Exemples

Dans ce graphe, il y a 4 arcs d'origine D. Leurs extrémités sont les points C,D,E et F.
Ainsi $\Gamma^+(D) = \{C; D; E; F\}$.



On peut retrouver le graphe à partir du tableau des successeurs.

sommet	A	B	C	D	E	F
successeurs	B	A, C, E	—	C, D, E, F	B	C

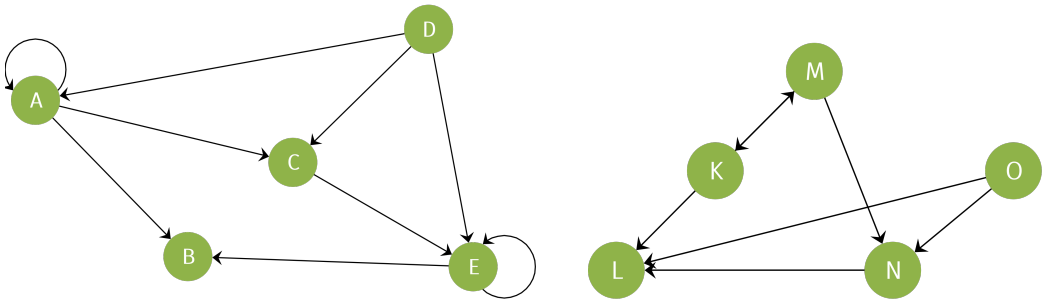


Dans ce graphe, il y a 3 arcs d'extrémité C.
Leurs origines sont les points B,D et F.
Ainsi $\Gamma^{-}(C) = \{B; D; F\}$. On peut retrouver le graphe à partir du tableau des prédécesseurs.

sommet	A	B	C	D	E	F
prédécesseurs	B	A, E	B, D, F	B, D	B, D	D

Exercice 116

Pour chacun des graphes, donne le tableau des successeurs et celui des prédécesseurs (attention : pour le graphe de gauche, on a dessiné des arcs bidirectionnels, chacun compte pour 2 arcs).



Exercice 117

Dessine le graphe correspondant au tableau de successeurs.

sommet	A	B	C	D
successeurs	A,B,C	B,C,D	C,D,A	D,A,B

Exercice 118

Dessine le graphe correspondant au tableau de prédécesseurs.

sommet	A	B	C	D
prédécesseurs	A,D	—	A,B,D	A

2.3 Matrice d'adjacence

Définition : matrice d'adjacence

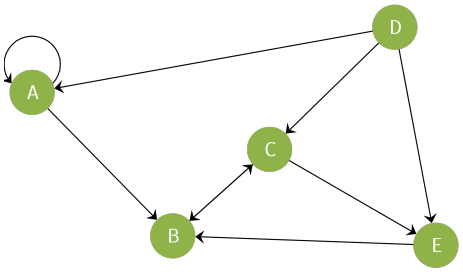
Soit un graphe G de sommets $S = \{s_1; ...; s_n\}$ et d'arcs A. On appelle *matrice d'adjacence de* G la matrice carrée d'ordre n $M = (m_{ij})$ telle que

- $m_{ij} = 1$ s'il y a un arc partant de s_i et arrivant sur s_j ;
- $m_{ij} = 0$ sinon.

Exemple

En écrivant les sommets dans l'ordre alphabétique, la matrice d'adjacence du graphe ci-dessous est

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \boxed{1} & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$



L'élément encadré de M est à la 3e ligne et à la 2e colonne : c'est $m32$. Il vaut 1 et signifie qu'il y a un arc partant du sommet 3, donc C, et allant au sommet 2, donc B.
De même, du 5e sommet E, il existe un arc allant vers le 4e (D), donc $m_{54} = 1$ (en rouge).

Remarque

Dans une matrice d'adjacence

- Les lignes correspondent au points de départ, les colonnes au point d'arrivée ;
- Sur une ligne donnée(la ie par exemple), on peut lire *tous les successeurs* de s_i ;
- Sur une colonne donnée, (la je par exemple), on lite *tous les prédécesseurs* de s_j ;

Exercice 119

On considère un graphe dont l'ensemble des sommets est {A; B; C; D; E; F}, et dont la matrice

d'adjacence est

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

1. Donner le tableau des successeurs de chaque sommet.
2. Donner le tableau des prédécesseurs de chaque sommet.
3. Représenter ce graphe.

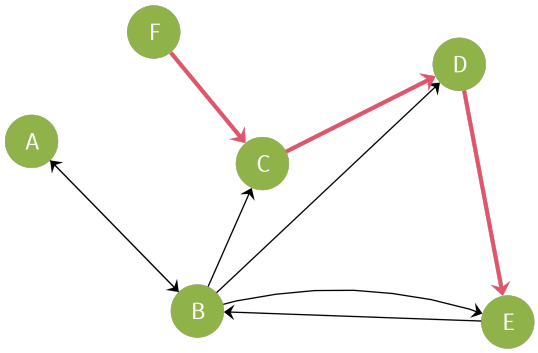
3 Chemins et circuits

Définitions : chemin, longueur

On considère un graphe orienté.
Un *chemin* est une succession de sommets dans un ordre donné, chacun étant relié au suivant par un arc.
La *longueur* du chemin, c'est le nombre d'arcs qui composent le chemin. C'est aussi le nombre de sommets qui composent le chemin mois un.

Exemple

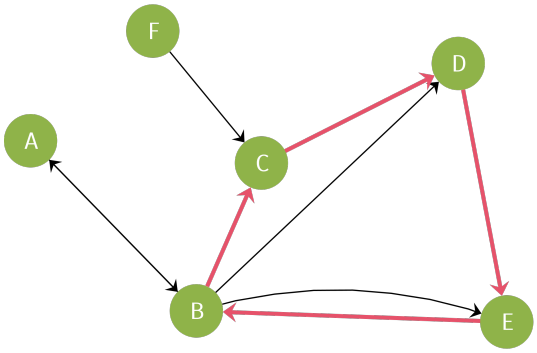
(F, C, D, E) est un chemin de longueur 3.
(F, C, B, E) n'en est pas un car l'arc (C, B) n'existe pas.



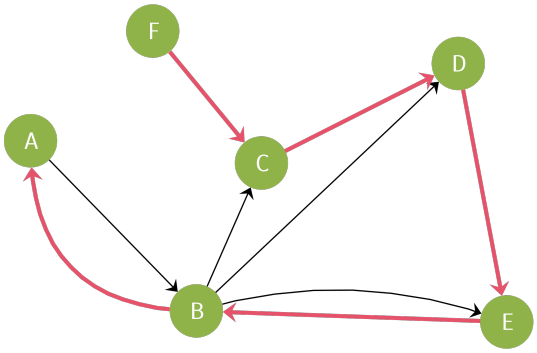
Définitions : chemin hamiltonien,circuit

Un *chemin hamiltonien* est un chemin qui passe une et une seule fois par *chaque* sommet.
Un *circuit* est un chemin dont le premier et le dernier sommet sont identiques (un chemin « fermé » en quelque sorte).

Exemples



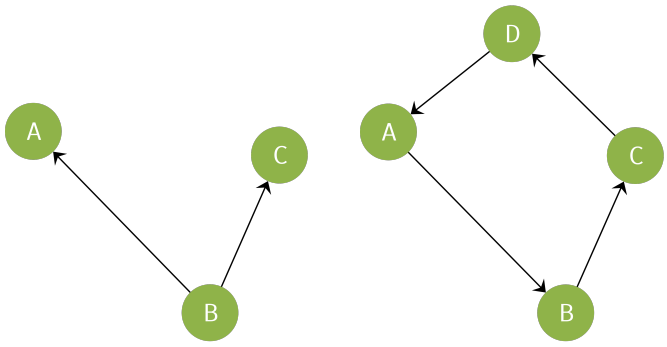
(C, D, E, B, C) est un circuit de longueur 4.



(F, C, D, E, B, A) est un chemin hamiltonien.

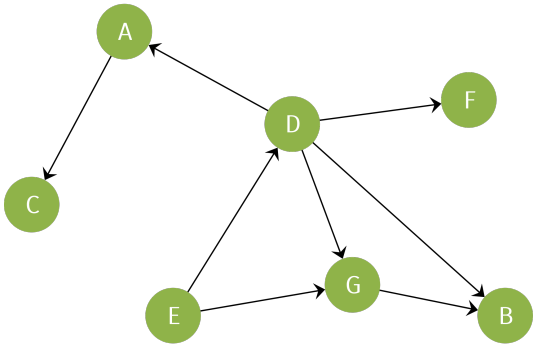
Remarque

Un graphe orienté *ne possède pas toujours* de circuit hamiltonien (à gauche) ou bien peut en posséder plusieurs (à droite).



Exercice 120

Trouve un chemin hamiltonien, un circuit de longueur 3 et un autre de longueur 4.



4 Utilité des matrices d'adjacence

On peut se poser beaucoup de questions ayant trait aux chemins d'un graphe. En voici trois que nous allons étudier :

- On se donne un graphe à 10 sommets, on en choisit un en particulier. Combien de chemins différents de longueur 5 commencent en ce sommet ?
- Toujours dans ce graphe, combien y a-t-il de chemin de longueur 5 ?
- Si on veut (comme dans l'exemple introductif) ajouter tous les « raccourcis » aux arcs du graphe, comment s'y prendre pour n'en oublier aucun ?

Propriété

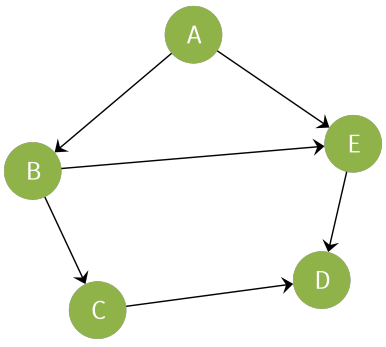
Soit G un graphe possédant n sommets s_1, s_2, ...s_n et M sa matrice d'adjacence.
Soit p un entier naturel positif. Alors M^p contient les informations sur les chemins de longueur p du graphe :
Le nombre de chemins de longueur p reliant s_i à s_j est le coefficient de la i^e ligne et de la j^e colonne de M^p.

Exemple

La matrice d'adjacence du graphe ci-contre est

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Pour déterminer les chemins de longueur 3, calculons M^3 *à l'aide de la calculatrice* :



$$M^3 = \begin{pmatrix} 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

et le seul coefficient non nul est à la 1re ligne (départ A) et à la 3e colonne (arrivée D) : il n'y a que 2 chemins de longueur 3 dans ce graphe, et ils relient A à D.
Maintenant qu'on connaît leur nombre et leurs extrémités, on peut les écrire : (A, B, E, D) et (A, B, C, D).

Remarque

Ce procédé ne donne pas la liste des chemins de longueur donnée, seulement leur nombre et leurs extrémités.

Exercice 121

On donne le tableau de prédécesseurs suivants :

sommet	A	B	C	D	E	F
prédécesseurs	—	—	A,B,E	F	B,D	A,B

En utilisant les puissances de la matrice d'adjacence :

- Donner tous les chemins de longueur 3.
- Montrer qu'il n'existe pas de chemin de longueur 4.
- Montrer que ce graphe ne possède pas de circuit.

5 Matrices booléennes et fermeture transitive

Parfois on n'a pas besoin de connaître le nombre précis de chemins d'une longueur donnée reliant deux sommets. On veut juste savoir s'il en existe au moins un ou pas. Les matrices booléennes vont nous permettre de répondre simplement à cette question.

La matrice d'adjacence d'un graphe ne comporte que des zéros et des uns donc on peut la voir comme une *matrice booléenne*, c'est à dire une matrice à coefficients dans l'algèbre de Boole $\mathcal{B} = \{0; 1\}$. Pour rappel, cette algèbre de Boole est munie des opérations binaires $+$ et $\times$ vérifiant :

- $0 + 0 = 0$, $1 + 0 = 1$ et $1 + 1 = 1$ (penser au « ou » logique)
- $0 \times 0 = 0$, $1 \times 0 = 0$ et $1 \times 1 = 1$ (penser au « et » logique)

Définitions : addition et multiplication de matrices booléennes

Soient A et B 2 matrices booléennes.

- On définit $A \oplus B$, somme *booléenne* de A et de B comme ceci : chaque coefficient de $A \oplus B$ est la somme booléenne des coefficients correspondants de A et de B.
  En pratique on calcule $A + B$ comme d'habitude et on remplace chaque coefficient non nul par un 1.

– On définit $A \otimes B$, produit *booléenne* de A et de B comme suit : on calcule $A \times B$ comme d'habitude et on remplace chaque coefficient non nul par un 1.

Exemples

Prenons $A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$.

– $A + B = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 0 & 2 \end{pmatrix}$ donc $A \oplus B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$

– $A \times B = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ donc $A \otimes B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

Définition : puissance d'une matrice booléenne

Soit A une matrice booléenne carrée de taille n.
On pose $A^{[0]} = I_n$, $A^{[1]} = A$ et pour tout entier p supérieur à 1 :

$$A^{[p]} = \underbrace{A \otimes \ldots \otimes A}_{p \text{ facteurs}}$$

En pratique il suffit de calculer A^p et de remplacer les coefficients non nuls par des 1.

Exemple

Prenons $A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ et $p = 3$.

Avec la calculatrice on obtient $A^3 = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$ donc $A^{[3]} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

Si A est la matrice d'adjacence d'un graphe alors $A^{[3]}$ nous indique si 2 sommets du graphe peuvent être reliés ou non par un chemin de longueur 3 : on voit que c'est toujours possible sauf pour relier le 1er au 3e .

On considère un graphe et on veut (comme dans l'exemple introductif) ajouter tous les « raccourcis » aux arcs du graphe, comment s'y prendre pour n'en oublier aucun ?
Si notre graphe possède n sommets, considérons 2 sommets s_i et s_j, on veut rajouter l'arc (s_i, s_j) (qu'on va appeler *raccourci*) aux arcs du graphe dès qu'il existe un chemin allant de s_i à s_j.
Nous allons admettre que c'est le cas si (et seulement si) il existe un chemin de longueur 1, ou 2, ..., ou $n-1$ qui relie ces 2 sommets. On arrive alors à la propriété et définition suivante :

Propriété et définition

Soit M la matrice d'adjacence d'un graphe G à n sommets. On définit

$$\widehat{M} = M \oplus M^{[2]} \oplus \ldots \oplus M^{[n-1]}$$

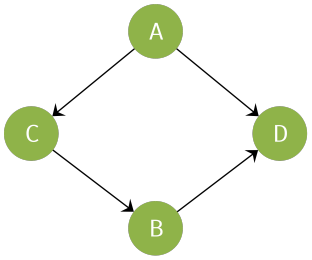
$\widehat{M}$ est la matrice d'adjacence du graphe appelé *fermeture transitive* de G, composé des mêmes sommets et arcs que ceux de G, auxquels on ajoute tous les arcs des « raccourcis » .

Méthode : déterminer une fermeture transitive

On se donne le graphe ci-dessous, dont la matrice d'adjacence est

$$M = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

et on aimerait déterminer sa fermeture transitive.



D'après la propriété précédente, étant donné que ce graphe possède 4 sommets on a

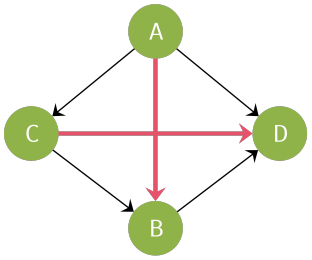
$$\widehat{M} = M \oplus M^{[2]} \oplus M^{[3]}$$

En pratique, on calcule M^2, M^3 puis $M + M^2 + M^3$ et on remplace chaque coefficient non nul par un 1.

Avec la calculatrice on obtient : $M^2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ et $M^3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ de sorte que

$$M + M^2 + M^3 = \begin{pmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

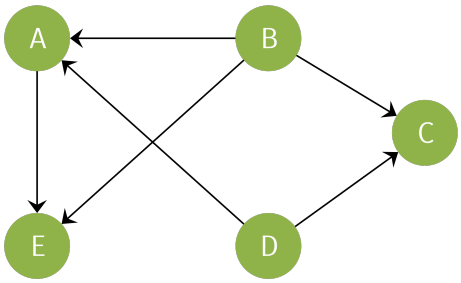
Donc en définitive $\widehat{M} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ et on aboutit au graphe suivant :



où l'on a fait figurer en rouge les « raccourcis » ajoutés. On peut remarquer que M^3 n'apporte pas grand-chose ici, car son seul coefficient non nul dit qu'il y a un chemin de longueur 3 : (A, C, B, D) reliant A et D, mais comme l'arc (A, D) existe déjà, « le raccourci est déjà là » .

Exercice 122

Voici un graphe.



1. Donner M, matrice d'adjacence du graphe.
2. Calculer $\widehat{M}$, matrice de la fermeture transitive de ce graphe.
3. Quels arcs doit-on ajouter au graphe ci-dessus pour réaliser sa fermeture transitive ?

6 Exercices

Exercice 123

Cinq joueurs, notés A, B, C, D et E, jouent régulièrement à un jeu en ligne. Chaque partie de ce jeu oppose deux adversaires. Le tableau suivant donne, pour chacun des cinq joueurs, la liste des adversaires qu'il a déjà battus.

Le joueur	a déjà battu
A	B, D
B	C
C	B, D
D	E
E	D

Ainsi, par exemple, le joueur C a déjà battu les joueurs B et D.

1. Graphe orienté associé à la situation

   a. En considérant le tableau précédent comme un tableau de successeurs, représenter la situation par un graphe orienté G, dans lequel un arc relie un sommet x à un sommet y si le joueur x a déjà battu le joueur y.

   b. Écrire la matrice d'adjacence M du graphe G.

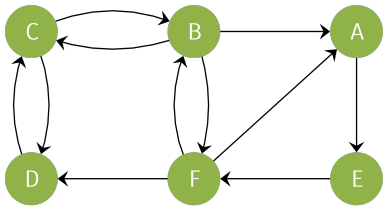
   c. Recopier et compléter le tableau des **prédécesseurs** dans le graphe G.

Le joueur	a déjà ...
A	
B	
C	
D	
E	

   d. Le graphe G contient-il un circuit ? Contient-il un chemin hamiltonien ? Justifier les réponses.

Exercice 124

Un fournisseur doit livrer 5 entreprises. Le réseau de transport est représenté par le graphe orienté donné ci-dessous où l'entrepôt du fournisseur est noté F, et les entreprises sont notées A, B, C, D, E.



1. Écrire la matrice d'adjacence M de ce graphe en considérant les sommets notés A, B, C, D, E, et F dans cet ordre.

2. Le fournisseur souhaite livrer chacune des entreprises. Il part de son entrepôt.

   a. Existe-t-il un chemin hamiltonien d'origine F dans ce graphe ? Si oui, citer un tel chemin.

   b. Interpréter le résultat relativement aux possibilités de livraison.

3. On donne la matrice $M^3 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 4 & 0 & 1 & 3 \\ 1 & 3 & 0 & 3 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 & 1 & 1 \\ 1 & 3 & 0 & 3 & 1 & 1 \end{pmatrix}$.

   a. Dans le contexte de l'exercice, interpréter le coefficient 2 situé sur la quatrième ligne et la troisième colonne de la matrice M^3.

   b. Combien existe-t-il de chemins de longueur 3 issus du sommet D dans ce graphe ? Justifier puis citer ces chemins.

   c. Le fournisseur doit maintenant effectuer une livraison, depuis l'entrepôt, dans quatre entreprises en commençant par l'entreprise D.

   Montrer que, pour effectuer cette livraison sans repasser par une entreprise déjà livrée, le fournisseur n'a qu'un seul chemin possible.

   Expliquer la démarche et préciser ce chemin.

Exercice 125

Partie A

Quatre sites internet traitent les changements climatiques et leurs conséquences sur la planète.

On considère une page web sur chacun de ces sites, et on note ces quatre pages A, B, C et D. Les liens hypertextes respectifs entre ces quatre pages sont tous récapitulés dans l'énumération suivante :

- A reçoit un unique lien de B et un unique lien de C ;
- B reçoit un unique lien de D ;
- C reçoit un unique lien de B, un unique lien de D et un unique lien de A ;
- D reçoit un unique lien de A.

1. Représenter l'ensemble de ces liens par un graphe orienté G de sommets A, B, C, D, dans lequel, si une page Y reçoit un lien d'une page X, on représente un arc du sommet X vers le sommet Y.

2. a. Donner la matrice d'adjacence M du graphe G.

   b. Interpréter les valeurs des termes situés sur la diagonale de la matrice M.

   c. Calculer la matrice M^4.

   d. Le graphe contient-il des circuits ? Justifier la réponse.

   e. Interpréter le terme de la 1re et 3e colonne de la matrice M^4 en termes de chemins, puis donner la liste de ces chemins.

3. Calculer $\widehat{M}$, la matrice de la fermeture transitive du graphe G.
   Interpréter le résultat obtenu dans le contexte de l'exercice.

Partie B

Un étudiant du BTS SIO a mis en place un moteur de recherche avec lequel les pages affichées sont ordonnées par pertinence, selon le nombre de liens hypertextes pointant vers chaque page.

Cette partie étudie un exemple simplifié, en limitant ce moteur de recherche aux quatre pages web A, B, C et D définies dans la partie A, et en considérant le graphe associé G.
La méthode mise en place par l'étudiant consiste à associer un score à chaque sommet du graphe.
Les scores a, b, c, d de chacun des sommets A, B, C, D, sont calculés à partir des instructions suivantes :

- on liste les prédécesseurs du sommet considéré dans le graphe G ;
- on divise le score de chaque prédécesseur par le nombre de ses successeurs ;
- le score d'un sommet est obtenu en ajoutant les quotients obtenus.

Exemple : le sommet A possède deux prédécesseurs B et C ; B a 2 successeurs et C a 1 successeur. D'où $a = \frac{b}{2} + \frac{c}{1}$.

1. Justifier l'égalité : $c = \frac{a}{2} + \frac{b}{2} + \frac{d}{2}$.
2. En établissant les quatre égalités vérifiées par les scores a, b, c, d, on obtient un système de quatre équations linéaires aux inconnues a, b, c, d. Ce système ayant une infinité de solutions, toutes proportionnelles entre elles, on pose $a = 1$ et on admet que la résolution se ramène à celle du système :

$$(\text{S}) \quad \begin{cases} 0,5b + c & = & 1 \\ b - 0,5d & = & 0 \\ 0,5b - c + 0,5d & = & -0,5 \end{cases}$$

On définit les matrices $\text{X} = \begin{pmatrix} b \\ c \\ d \end{pmatrix}$, $\text{A} = \begin{pmatrix} 0,5 & 1 & 0 \\ 1 & 0 & -0,5 \\ 0,5 & -1 & 0,5 \end{pmatrix}$

et $\text{B} = \begin{pmatrix} 0,5 & 0,5 & 0,5 \\ 0,75 & -0,25 & -0,25 \\ 1 & -1 & 1 \end{pmatrix}$.

   a. Exprimer le système (S) sous la forme $\text{A} \times \text{X} = \text{Y}$, où Y est une matrice à préciser.
   b. Calculer le produit $\text{B} \times \text{A}$.
   c. En déduire que $\text{X} = \text{B} \times \text{Y}$, puis donner la solution du système (S).

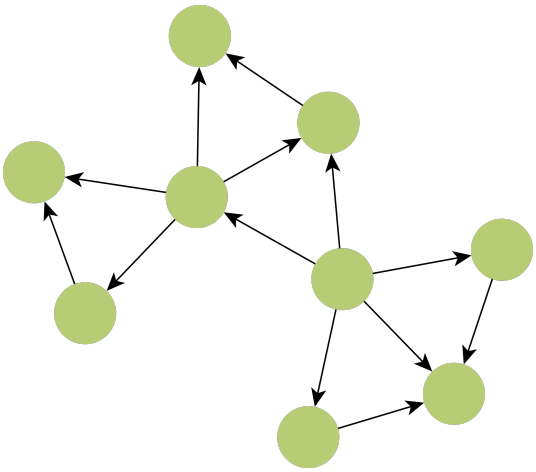
3. Donner, en justifiant, le classement des pages web A, B, C et D selon la méthode mise en place.

Chapitre 9

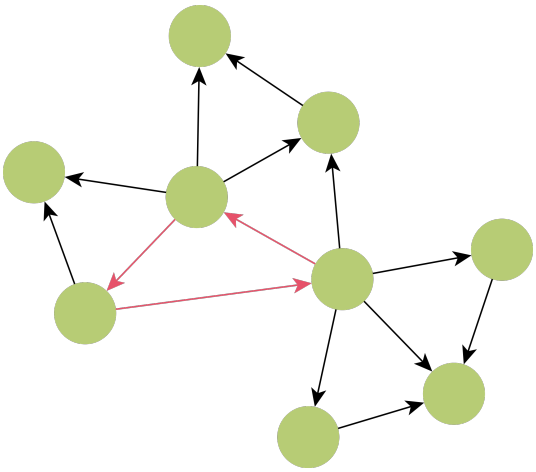
Graphes : méthodes et algorithmes

1 Niveaux dans un graphe orienté sans circuit

On dit qu'un graphe orienté est *sans circuit* lorsqu'il ...ne comporte aucun circuit, c'est-à-dire qu'on ne peut pas trouver un chemin partant d'un sommet et y revenant.

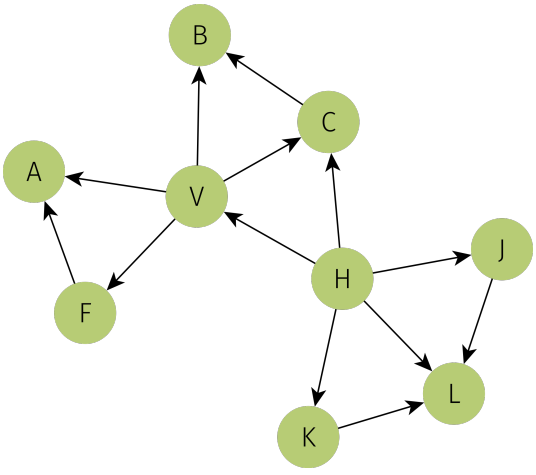


graphe sans circuit

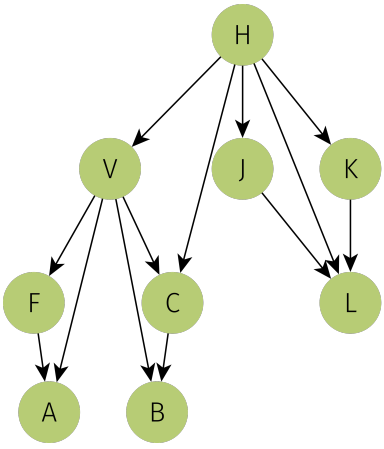


graphe avec circuit

Lorsqu'un graphe orienté est sans circuit il est possible de l'organiser de manière *hiérarchisée*, comme ceci :



graphe sans circuit



le même graphe hiérarchisé

Comment faire ? À droite, on voit que les sommets sont disposés en couches horizontales : des *niveaux*. Il faut donc définir ce qu'est le niveau d'un sommet.

Propriété

Dans un graphe orienté sans circuit, il existe un sommet qui n'a pas de prédécesseur.

Preuve

Soit n le nombre de sommets du graphe. On va raisonner par l'absurde : supposons que notre graphe soit sans circuit, mais qu'il n'existe aucun sommet sans prédécesseur. Alors tout sommet possède un prédécesseur.
On en choisit un, on l'appelle S_0 puis un de ses prédécesseurs qu' on appelle S_1

- si $S_1 = S_0$ alors il y a une boucle (donc un circuit de longueur 1) sur S_0 et c'est contradictoire avec notre hypothèse.
- sinon on continue et on appelle S_2 un prédécesseur de S_1.

À chaque nouveau sommet choisi si c'est un sommet qui figure déjà dans la liste des sommets choisis, cela nous permet d'exhiber un circuit et c'est contradictoire. Or comme il n'y a que n sommets, au bout de n étapes (au maximum), on sera obligés de choisir un sommet déjà choisi.

Définition : niveau d'un sommet dans un graphe orienté sans circuit

Soit un sommet d'un graphe orienté sans circuit

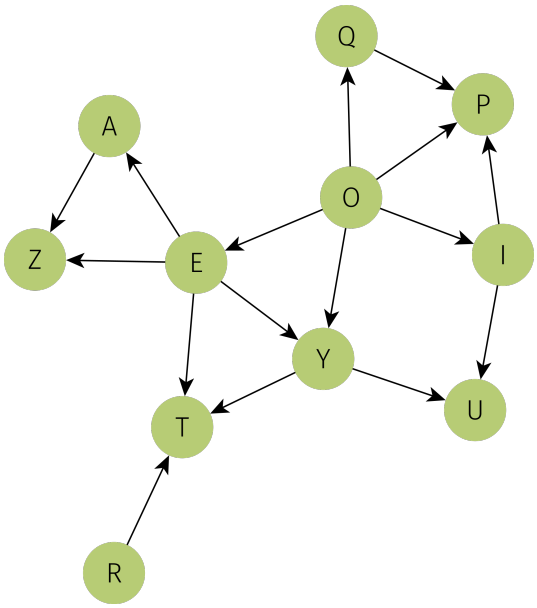
- s'il n'a pas de prédécesseur, son niveau est 0 ;
- sinon, on considère tous ses prédécesseurs, on choisit celui qui a le niveau le plus élevé et on ajoute 1 : on obtient le niveau du sommet.

Pour déterminer les niveaux des sommets on peut aussi appliquer l'algorithme suivant :

```
Variables
    L : liste des sommets
    n : entier
Début
    n <-- 0
    tant que L est non vide
        sélectionner tous les sommets qui n'ont aucun prédécesseur dans L
        ils ont le niveau n
        enlever ces sommets de L
    n <-- n + 1
Fin
```

Exemple

On considère le graphe suivant :



On commence par construire le tableau des prédécesseurs :

sommet	A	Z	E	R	T	Y	U	I	O	P	Q
prédécesseurs	E	A,E	O	—	R,E,Y	E,O	Y,I	O	—	I,O,Q	O

R et O ont le niveau 0. On les retire du tableau :

sommet	A	Z	E	T	Y	U	I	P	Q
prédécesseurs	E	A,E	—	E,Y	E	Y,I	—	I,Q	—

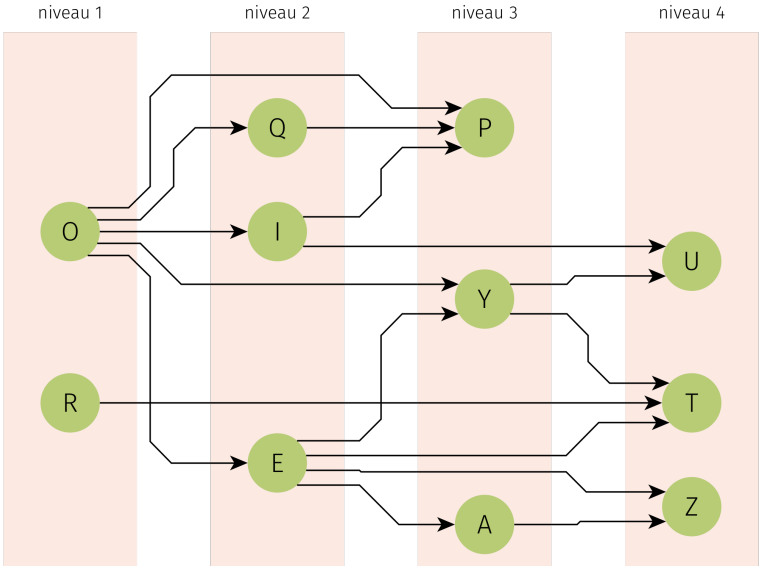
E, I et Q ont le niveau 1, on les retire :

sommet	A	Z	T	Y	U	P
prédécesseurs	—	A	Y	—	Y	—

A, Y et P ont le niveau 2 et on voit que Z, T et U ont le niveau 3.

sommet	R	O	E	I	Q	A	Y	P	Z	T	U
niveau	0	0	1	1	1	2	2	2	3	3	3

On peut maintenant redessiner le graphe de manière hiérarchisée de haut en bas ou de droite à gauche, en alignant les sommets par niveaux :

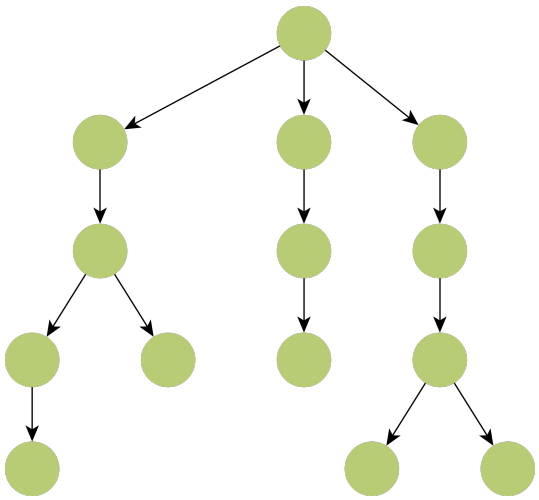


graphe représenté de manière nivelée

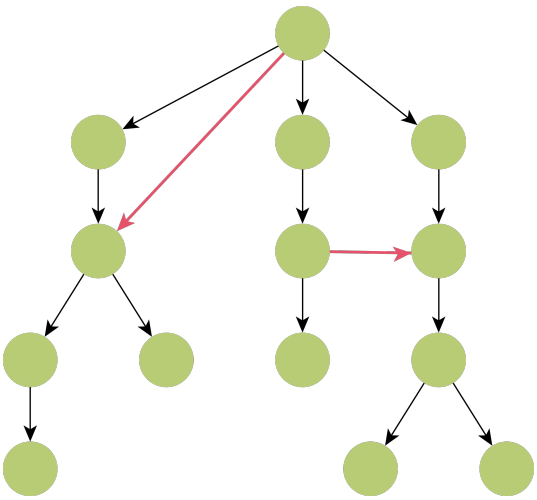
En appliquant cette méthode on peut parfois prouver qu'un graphe donné est une *arborescence*.

Définition : arborescence

Une *arborescence* est un graphe oriente qui possède *un unique sommet de niveau 0*, qu'on appelle la *racine* et à partir de laquelle on peut atteindre *tout autre sommet par un unique chemin*.

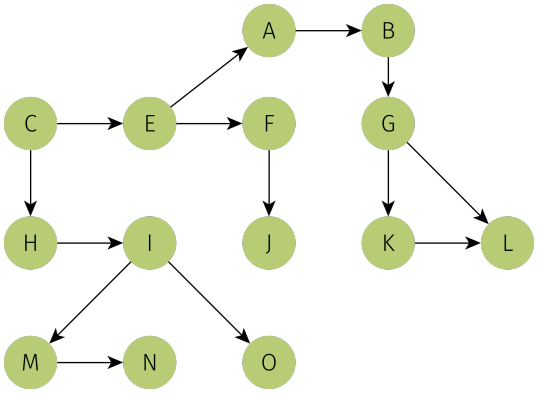


graphe qui est une arborescence



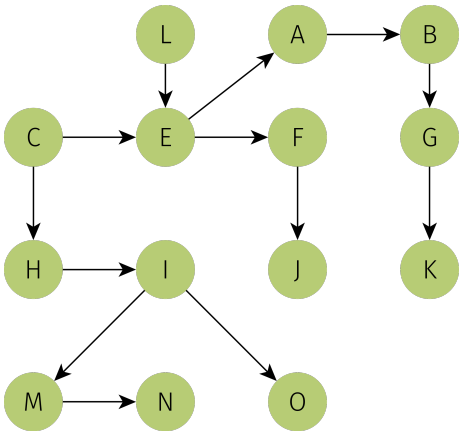
graphe qui n'en est pas une

Exercice 126



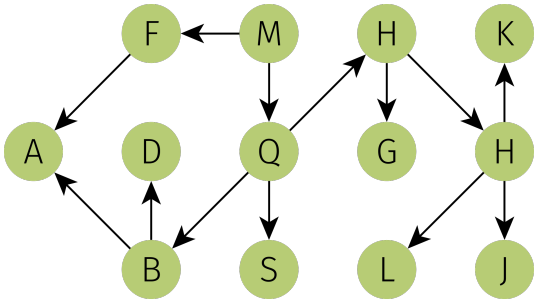
1. Dresser le tableau des prédécesseurs du graphe ci-dessus.
2. Déterminer le niveau de chaque sommet.
3. Dessiner le graphe de manière hiérarchisée.
4. Ce graphe est-il une arborescence ?

Exercice 127



Même consigne.

Exercice 128

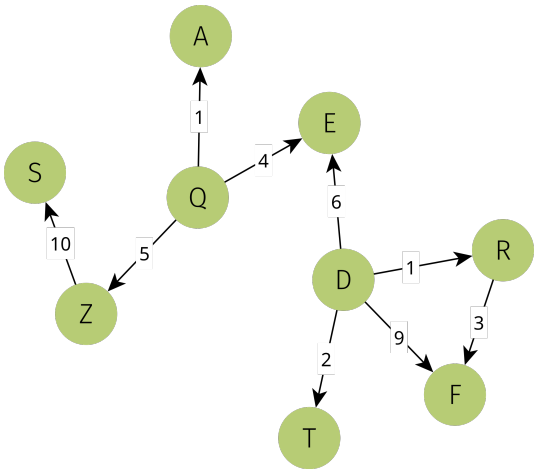


Même consigne.

2 Graphes orientés valués et ordonnancement

Définition : graphe orienté valué

un graphe orienté est dit *valué* (on dit aussi pondéré) lorsqu'on attribue un nombre réel à chacun de ses arcs.



un exemple de graphe valué

Imaginons maintenant une équipe de plusieurs personnes qui travaille sur un projet, (ce peut être le développement d'une application ou bien la construction d'une salle de sport). Toutes les personnes ne travaillent pas sur les mêmes aspects du projet. Certaines tâches peuvent être réalisées en même temps (par exemple on peut penser qu'on peut poser un revêtement au sol de la salle de sport en même temps qu'on peint sa façade). Certaines doivent attendre que d'autres soient terminées pour pouvoir commencer (évidemment avant de peindre la façade il faut avoir monté la façade), on parle de *contraintes d'antériorité*.
Les contraintes d'antériorité conduisent naturellement à un graphe orienté. Chaque tâche possède une durée propre, qui conduit à pondérer le graphe.

Faire de l'ordonnancement, c'est préciser la chronologie des différentes tâches (éventuellement effectuées en parallèle), déterminer la durée minimale du projet et les conséquences éventuelles d'un retard lors de la réalisation de telle ou telle tâche.

La méthode MPM sur un exemple

Cette méthode de gestion de projet, appelée méthode des potentiels métra (MPM) fut développée par le mathématicien français Bernard Roy en 1958, pour être directement appliquée en usine.

On va considérer le tableau de tâches suivant

Tâche	A	B	C	D	E	F	G
Durée en jours	6	3	6	2	4	3	1
Tâches antérieures	—	—	—	B	B	A, D	C, E, F

Dans celui-ci on lit, par exemple, que la tâche F dure 3 jours et ne peut commencer que lorsque A et D sont terminées.

Cet exemple va nous servir à illustrer la méthode MPM, mais nous donnerons les définitions dans le cas général.

Notations

On note G le graphe orienté valué qui représente les tâches du projet.
Il a n sommets s_1, ..., s_n qui représentent toutes les tâches. On écrit s_i pour parler d'un de ces sommets sans dire précisément lequel.
On note aussi $d(s_i)$ la durée de la tâche s_i, c'est la valeur (le poids) de tous les arcs qui partent de

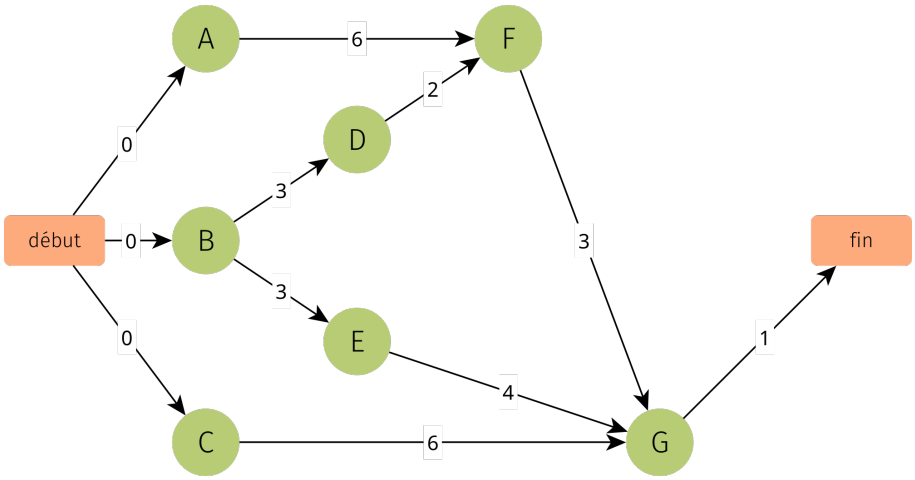
s_i.

Niveler le graphe

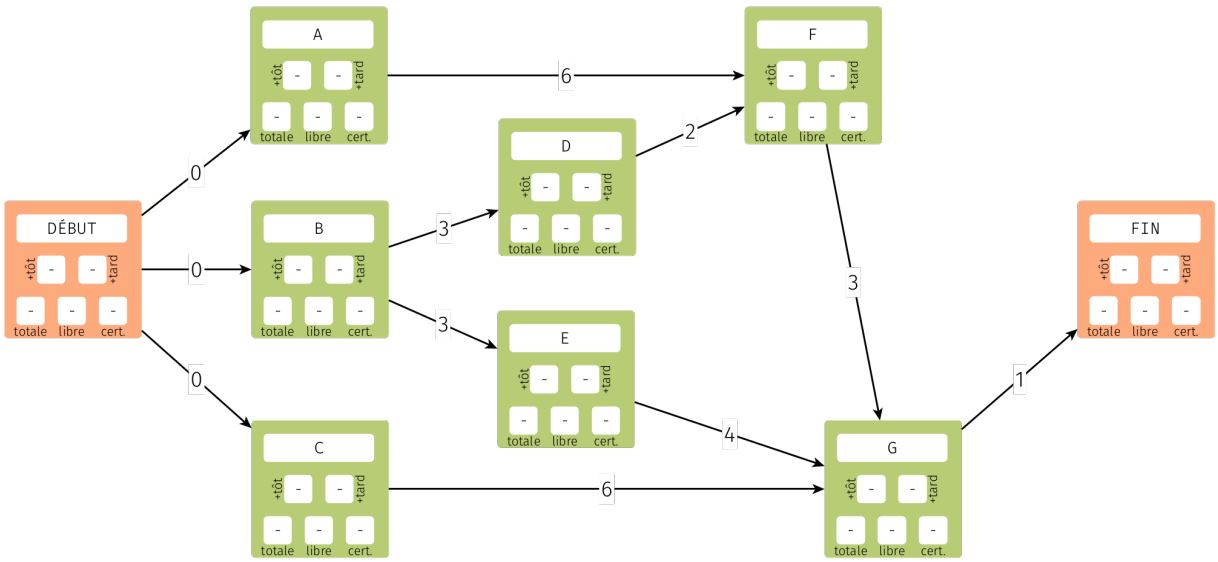
Dans le graphe qu'on va construire, les tâches antérieures sont les *prédécesseurs directs* de chaque tâche, et il n'y a pas de circuit (heureusement pour le projet). On peut donc niveler le graphe :

- A, B et C sont clairement de niveau 0 ;
- D et E sont de niveau 1 ;
- F est de niveau 2 ;
- G est de niveau 3.

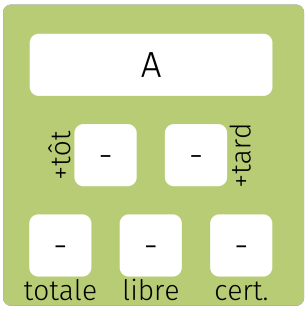
Cela nous permet de représenter le graphe du projet de manière hiérarchisée. On parle alors de *graphe d'ordonnancement*. Pour les besoins on rajoute 2 sommets « fictifs » : le début et la fin du projet. On pondère les arcs par les durées des tâches (on met 0 en partant de début car on considère que le projet peut commencer maintenant) :



Puis, étant donné que l'on va déterminer beaucoup de paramètres, on utilisera plutôt cette représentation :



Comme on peut le voir, pour chaque sommet, on peut déterminer 5 paramètres :



1. D'abord, on détermine la *date au plus tôt* de chaque tâche.
2. Ensuite on peut déterminer la *date au plus tard* de chaque tâche.
3. Cela permet de déterminer la *marge totale* de chaque tâche, ainsi que la *marge libre* et la *marge certaine*.

Nous allons définir ces nombres au fur et à mesure.

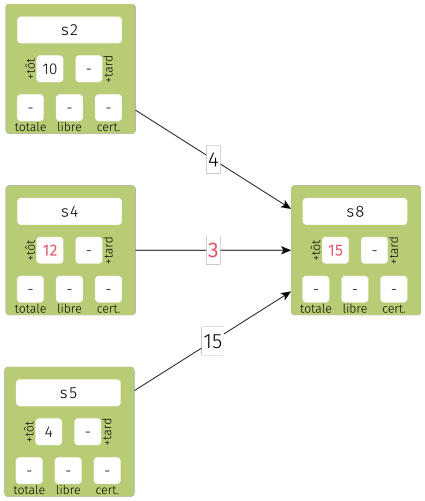
Date au plus tôt

Définition : date au plus tôt

On note $t(s_i)$ la *date au plus tôt* de la tâche s_i. Puisque s_i ne peut commencer que lorsque toutes les tâches antérieures sont terminées on a

$$t(s_i) = \max\left(\left\{t(s_j) + d(s_j) \ : \ s_j \text{ prédécesseur de } s_i\right\}\right)$$

Exemple

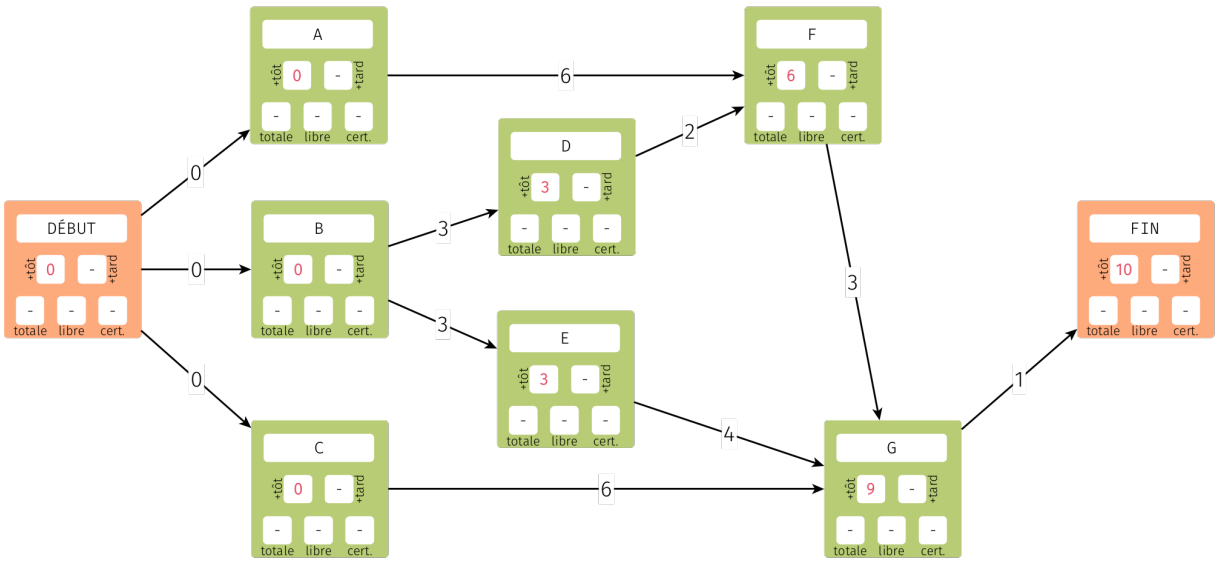


Pour calculer la date au plus tôt de s_8, il faut que les tâches antérieures s_2, s_4 et s_5 soient terminées :

- s_2 commence au bout de 10 jours, dure 4 jours, donc est terminée au bout de 14 jours ;
- s_4 commence au bout de 12 jours, dure 3 jours, donc est terminée au bout de 15 jours ;
- s_5 commence au bout de 4 jours, dure 10 jours, donc est terminée au bout de 14 jours.

En définitive, $t(s_8) = 15$.

Calculons toutes les dates au plus tôt de notre projet. On commence par le début et on procède *niveau par niveau* dans l'ordre hiérarchique :



Définition : durée minimale de réalisation d'un projet

C'est le temps minimal requis pour que toutes les tâches soient accomplies en respectant les contraintes.

Dans notre cas la *durée minimale du projet* est de 10 jours.

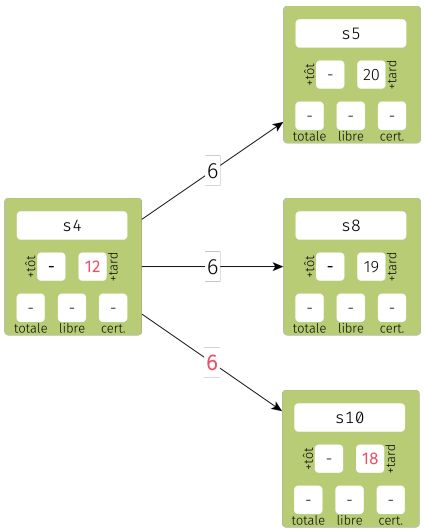
Date au plus tard

Définition : date au plus tard

On note $T(s_i)$ la *date au plus tard* de la tâche s_i : c'est la date maximale à laquelle on peut commencer s_i sans que cela ne repousse la date de fin du projet.

$$T(s_i) = \min\left(\left\{T(s_j) - d(s_i) \ : \ s_j \text{ successeur de } s_i\right\}\right)$$

Exemple

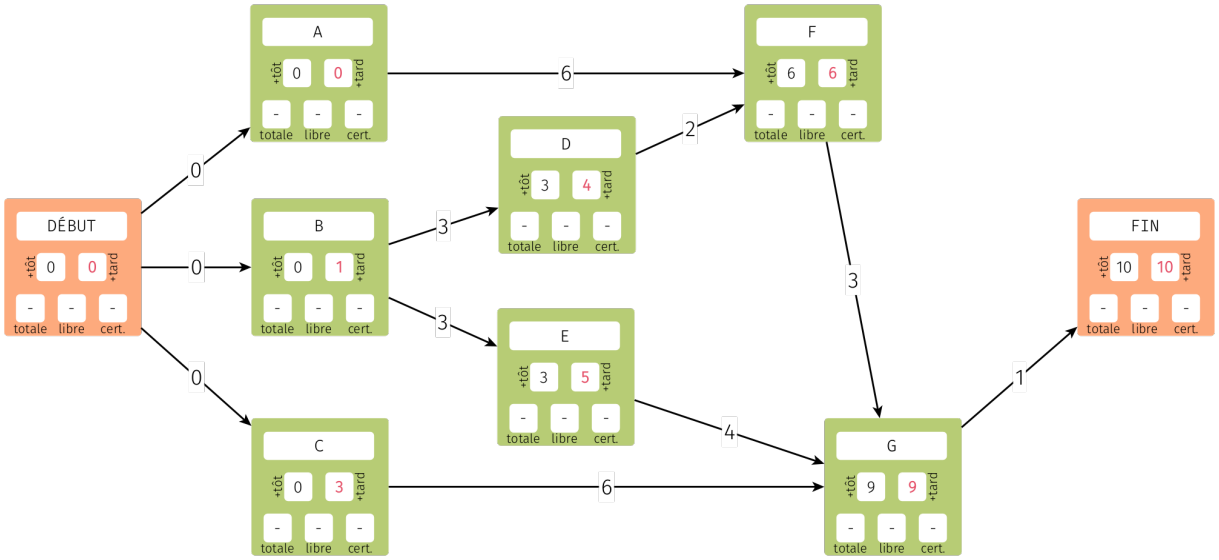


On n'a pas renseigné les dates au plus tôt car elles n'interviennent pas ici, mais normalement elles doivent avoir été calculées d'abord.
La tâche s_4 dure 6 jours

- puisqu'on peut commencer s_5 au plus tard au bout de 20 jours, il faut commencer s_4 au plus tard au bout de $20 - 6 = 14$ jours pour ne pas ralentir ;
- Puisqu'on peut commencer s_8 au plus tard au bout de 19 jours, il faut commencer s_4 au plus tard au bout de $19 - 6 = 13$ jours pour ne pas ralentir ;
- Puisqu'on peut commencer s_{10} au plus tard au bout de 18 jours, il faut commencer s_4 au plus tard au bout de $18 - 6 = 12$ jours pour ne pas ralentir.

Ainsi $T(s_4) = 12$.

Calculons toutes les dates au plus tard de notre projet. On commence par la fin : la date au plus tard de la fin est par définition égale à sa date au plus tôt. Ensuite on « remonte » la hiérarchie niveau par niveau :



On s'aperçoit que certaines tâches autorisent une « marge de manœuvre » et d'autres non. Nous allons préciser cela.

Marge totale, tâche critique

Définitions : marge totale et tâche critique

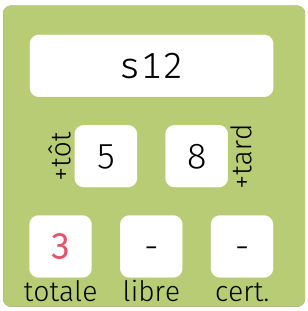
La *marge totale* d'une tâche s_i est

$$MT(s_i) = T(s_i) - t(s_i)$$

C'est le retard maximum que l'on peut accepter sur la date de début au plus tôt de la tâche sans que cela ne retarde la date de fin de projet.

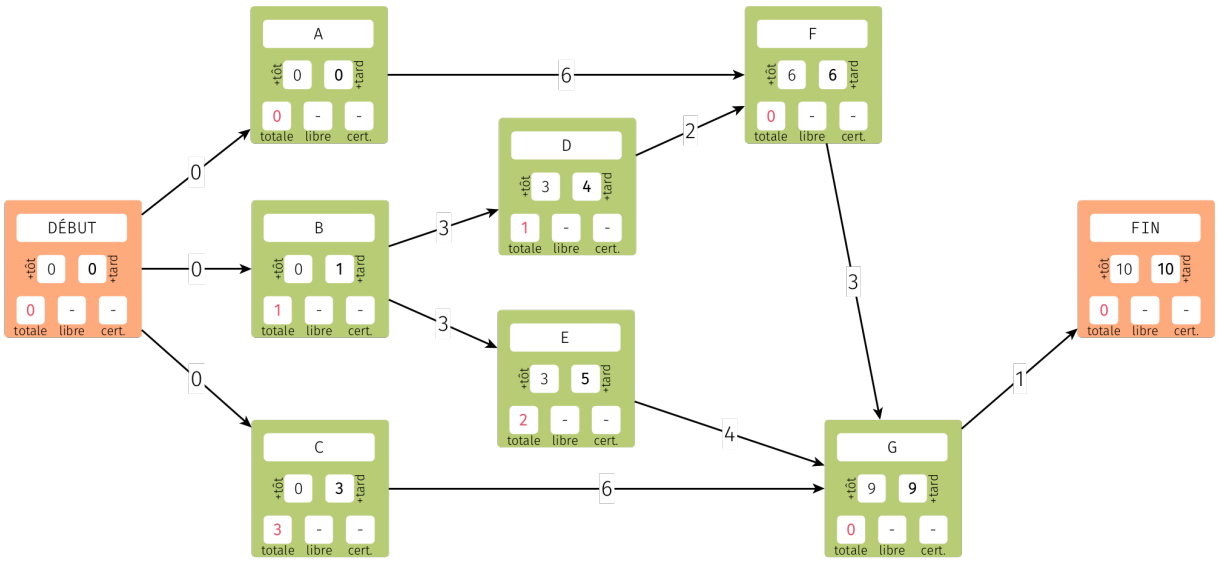
Si la marge totale d'une tâche est nulle, on dit que c'est une *tâche critique* : on doit impérativement la commencer au plus tôt, sinon le projet sera ralenti.

Exemple



La tâche s_{12} peut commencer au plus tôt au bout de 5 jours et au plus tard au bout de 8 jours. Sa marge totale est donc $\text{MT}(s_{12}) = 3$. Ce n'est pas une tâche critique : une fois que toutes les tâches antérieures à s_{12} sont effectuées, on peut en théorie encore attendre 3 semaines avant de commencer s_{12}.

Calculons les marges totales des tâches de notre projet :



Il existe des tâches critiques (en plus du début et de la fin) : A, F et G.

Définition : chemin critique

Un chemin critique est un chemin qui n'est composé que de tâches critiques.

Le chemin début-A-F-G-fin n'est composé que de tâches critiques : c'est un *chemin critique*.

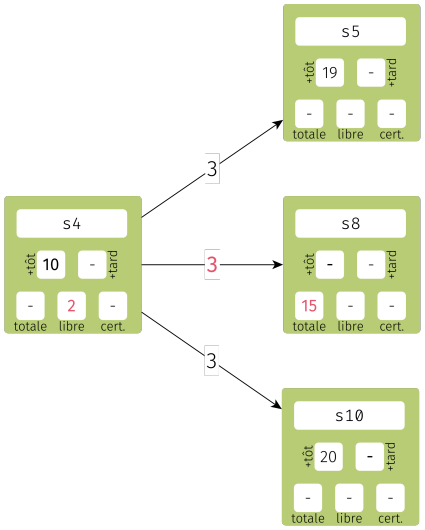
Marge libre

Définition : marge libre

La *marge libre* d'une tâche, c'est le retard maximum que l'on peut accepter sur la date de début *au plus tôt* d'une tâche sans que cela ne retarde la date de début *au plus tôt* de chacune des dates suivantes.

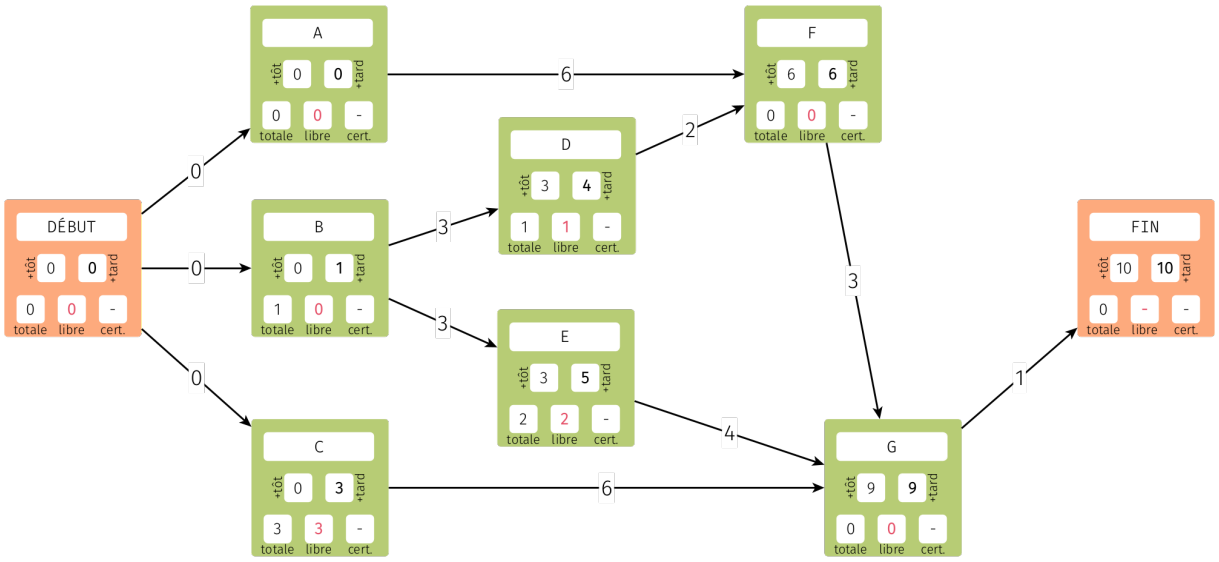
$$\text{ML}(s_i) = \min\left(\left\{t(s_j) - t(s_i) - d(s_i) \ : \ s_j \text{ successeur de } s_i\right\}\right)$$

Exemple



La tâche s_4 dure trois jours. En la commençant au plus tôt on finit à 13 jours. Il y a des marges pour chacune des tâches suivantes, la plus petite est pour s_8 et vaut 3 jours.
Ainsi $\mathrm{ML}(s_4) = 3$.

Calculons les marges libres pour notre exemple :



Marge certaine

Définition : marge certaine

La *marge certaine* d'une tâche est le retard que l'on peut prendre sur cette tâche sans modifier les dates de début au plus tôt des tâches postérieures, tout en sachant que les tâches précédentes ont commencées à leur date au plus tard.
Il faut commencer par calculer *la date au plus tôt retardée de la tâche*, c'est-à-dire la date au plus tôt de la tâche sachant que les tâches précédentes ont commencé à leur date au plus tard. Pour une tâche s_i cette date est

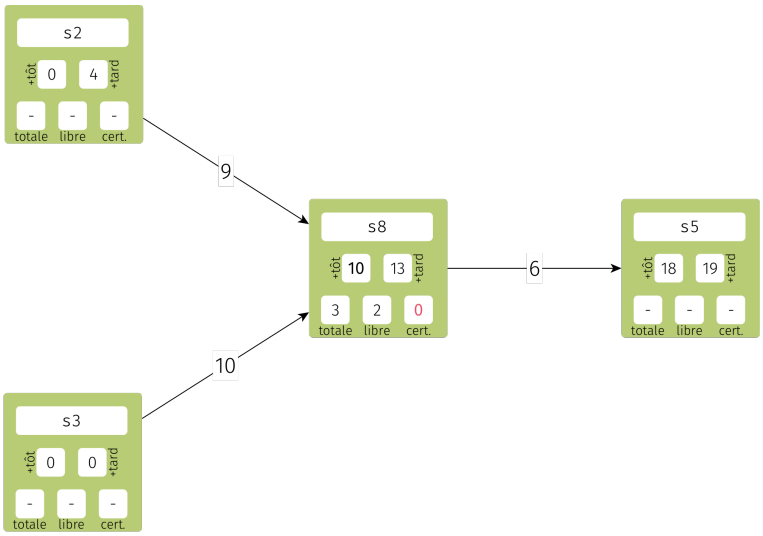
$$\mathrm{DR}(s_i) = \max\left(\left\{\mathrm{T}(s_j) + d(s_j) \ : \ s_j \text{ prédécesseur de } s_i\right\}\right)$$

Et alors la marge certaine est

$$\mathrm{MC}(s_i) = \min\left(\left\{t(s_j) - \mathrm{DR}(s_i) - d(s_i) \ : \ s_j \text{ successeur de } s_i\right\}\right)$$

Avec la convention que si ce résultat est négatif on décide que $\mathrm{MC}(s_i) = 0$.

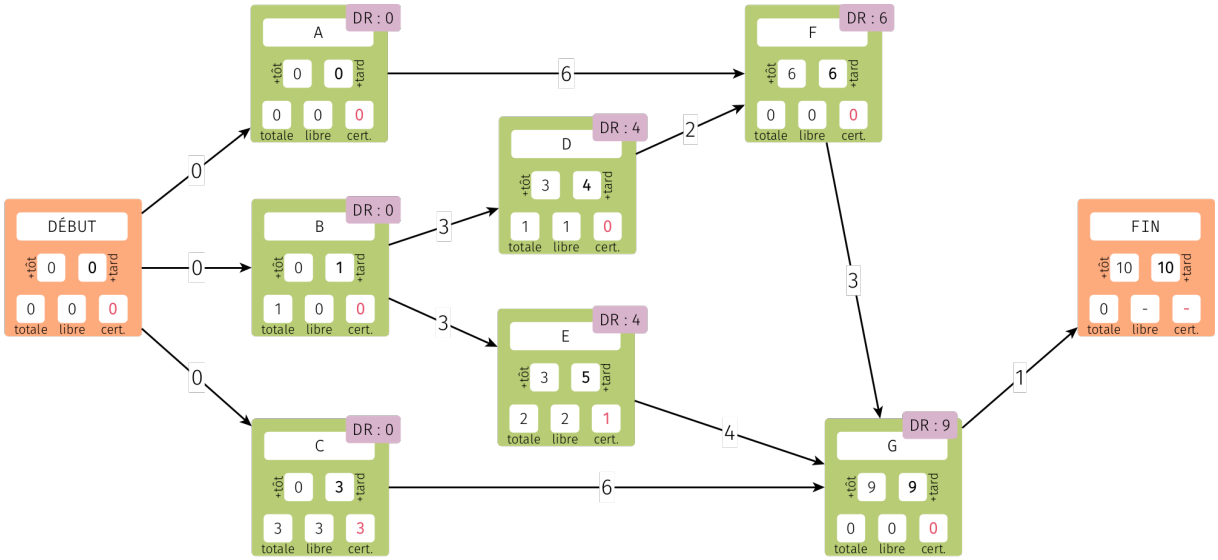
Exemple



Pour calculer la marge certaine de la tâche s_4 :

- On calcule $\mathrm{DR}(s_4)$: au pire s_3 commence à 0 et dure 10 donc fait commencer s_4 à 10, et au pire s_2 commence à 9 et dure 4 donc fait commencer s_4 à 13. La date retardée $\mathrm{DR}(s_4)$ vaut donc 13.
- s_4 n'a qu'un successeur : s_5, on calcule donc $t(s_5) - \mathrm{DR}(s_4) - d(s_4)$, cela donne $18 - 13 - 6 = -1$, c'est négatif donc $\mathrm{MC}(s_4) = 0$.

Calculons les marges certaines pour notre exemple :



L'intérêt de l'ordonnancement est de déterminer les tâches critiques, dont le déroulement devra être rigoureusement suivi pour ne pas perturber la suite du projet. Lorsque des tâches fortement non critiques (comme les tâches C ou E) sont trouvées, on peut par exemple diminuer un peu les ressources attribuées à ces tâches, quitte à utiliser leur marge libre, dans le but de réduire le coût global du projet.

Exercice 129

La mise en service d'un nouvel équipement routier demande la réalisation d'un certain nombre de tâches. Le tableau ci-dessous les recense, avec les contraintes d'antériorité.

Tâches	A	B	C	D	E	F	G
Durée en jours	6	3	6	2	4	3	1
Tâches antérieures	-	-	-	B	B	A, D	C, E, F

1. Déterminer le niveau de chacune des tâches.
2. Construire le graphe d'ordonnancement du projet et calculer les dates au plus tôt et au plus tard de chaque tâche.
3. Déterminer le chemin critique. Quelle est la durée minimale de réalisation du projet ?
4. Calculer la marge totale de la tâche E. Quelle est sa signification ?
5. Calculer la marge libre de C. Quelle est sa signification ?

Exercice 130

La réalisation d'un projet nécessite plusieurs tâches dont les duréees en jours et les contraintes d'antériorités sont résumées ci-dessous.

Tâches	A	B	C	D	E	F	G	H	I	J
Durée en jours	4	2	2	1	2	5	3	3	3	4
Tâches antérieures	-	-	A	A	A, B	C	D, E	E, G	H	F, I

1. Déterminer le niveau de chaque tâche.
2. Construire le graphe d'ordonnancement du projet et calculer les dates au plus tôt et au plus tard de chaque tâche.
3. Déterminer le chemin critique. Quelle est la durée minimale de réalisation du projet ?
4. En réalité, la tâche C a nécessité une durée de 5 jours. Est-ce que cela a eu une incidence sur la durée de réalisation du projet ?

Partie II

Programmation avec Python

Chapitre 10

Valeurs et types

1 Python, machine à évaluer

PYTHON est tout d'abord une machine à calculer, ou plutôt une machine à *évaluer* : lorsque PYTHON rencontre une *expression*, c'est-à-dire une écriture qui produit une valeur, il commence par déterminer cette valeur.
Pour s'en rendre compte, il suffit d'écrire des expressions dans une *console*.

Python

```
>>> 5 - 3
2

>>> 11 / (1 + 2)
3.6666666666666665

>>> 30 / 15
2.0
```

On se rend compte que les valeurs rencontrées ne sont pas présentées de la même manière : `5 - 3` a la valeur `2` alors que `30 / 15` a la valeur `2.0`.
Pour y voir plus clair, on peut appeler la fonction `type` qui

- en entrée prend une expression ;
- renvoie le *type* de l'expression.

Python

```
>>> type(2)
<class int>

>>> type(2.0)
<class float>
```

Il y a donc au moins deux types de valeurs, le type `int` et le type `float`.
En fait il existe une multitude de types prédéfinis selon la nature de la valeur à représenter et nous allons les passer en revue.

2 Le type int

Il sert à représenter les *entiers relatifs* (*integer* signifie « entier » en Anglais).
Le type `int` dispose des opérations `+` (addition), `-` (soustraction) et `*` (multiplication).

Python

```
>>> 3 + 2
5

>>> 2 * 3
6

>>> 3 - 2 * 2
-1

>>> 10_000 # on utilise des _ pour séparer les chiffres
```

On dispose également de *deux opérations très pratiques* : soient `a` et `b` deux `int`, et `b` non nul, alors on

- `a // b` est le *quotient* de la *division euclidienne* de `a` par `b` ;
- `a % b` est le *reste*.

Python

```
>>> 64 // 10 # 64, c'est 6 * 10 + 4
6

>>> 64 % 10
4

>>> 22 // 7 # 22, c'est 3 * 7 + 1
3

>>> 22 % 7
1
```

On dispose de l'opération d'*exponentiation* (opération puissance), notée `**`.
Attention : Cette opération peut produire un résultat *non-entier*, de type `float` (voir partie suivante).

Python

```
>>> 2 ** 3
8

>>> 10 ** 4
10000

>>> 2 ** (-1)
0.5
```

Pour finir, la *division décimale* peut être effectuée sur des entiers, mais elle renvoie un résultat de type `float`.

3 Le type float

Il sert à représenter les *nombres à virgule flottante* (*to float* : flotter en Anglais). Ce sont (en gros) des nombres décimaux. PYTHON comprend et utilise la notation scientifique : `2.35e6` vaut $2,35 \times 10^6$, c'est-à-dire 2 350 000.

Python

```
>>>  2 / 7
0.2857142857142857 # c'est une valeur approchée

>>> 1 / 100_000
1e-05

>>> 1.2e-4
0.00012
```

On peut pratiquer sur les `float` toutes les opérations vues avec les `int`. Pour des fonctions plus compliquées telles le cosinus ou l'exponentielle, on fait appel au module[1] `math` :

Python

```
>>> from math import *
>>> pi
3.141592653589793

>>> cos(pi / 3)
0.5000000000000001

>>> exp(2)
7.38905609893065

>>> log(2)
0.6931471805599453

>>> exp(log(2))
2.0
```

`exp` est la *fonction exponentielle* et `log` la *fonction logarithme népérien*[2] notée ln en France.

4 Le type str

Il sert à représenter les *chaînes de caractères* (`str` est l'abréviation de *string*, qui veut dire chaîne en anglais). Lorsqu'on écrit une valeur de type `str`, on peut utiliser les symboles `'`, `"` ou même `'''` (suivant que la chaîne contient des apostrophes, ou des guillemets).

Python

```
>>> 'Bonjour.'
'Bonjour'
```

[1] Un module est un ensemble de *fonctions* et/ou de *constantes* que l'on peut importer.
[2] Voir le programme de mathématiques de terminale scientifique.

```
>>> 'J'aime Python.'
SyntaxError

>>> "J'aime Python."
"J'aime Python."

>>> "Je n'aime pas qu'on m'appelle "geek"."
SyntaxError

>>> """Je n'aime pas qu'on m'appelle "geek"."""
'Je n\'aime pas qu\'on m\'appelle "geek".'
```

La dernière évaluation produit une valeur correcte. PYTHON utilise simplement `\'` pour écrire les apostrophes qui sont à l'intérieur de la valeur.

Le symbole + sert à *concaténer 2 chaînes*, c'est-à-dire à les mettre bout à bout.

Python

```
>>> 'Yes' + 'No'
'YesNo'

>>> 'No' + 'Yes'
'NoYes'
```

On peut même multiplier un `str` par un `int` :

Python

```
>>> 3*'Aïe ! '
'Aïe ! Aïe ! Aïe ! '
```

PYTHON évalue `3*'Aïe !'` comme `'Aïe ! ' + 'Aïe ! ' + 'Aïe ! '`.
Voici deux types très utiles que nous étudierons en détail plus tard.

5 Le type list

Une valeur de type `list` est une... liste ordonnée de valeurs. Celles-ci peuvent être du même type ou non.

Python

```
>>> [] # liste vide
[]

>>> [1, 4, 5] # liste comportant 3 int
[1, 4, 5]

>>> [2.0, -4, 'Bonjour', 'Coucou']
[2.0, -4, 'Bonjour', 'Coucou']
```

L'intérêt de ce type est de rassembler plusieurs valeurs au sein d'une seule, qui pourra ensuite être *parcourue*.

6 Le type dict

Celui-ci sert à établir des *association* du type *clé : valeur*.

Python

```
>>> {} # dictionnaire vide
{}

>>> {'France': 'Paris', 'Canada': 'Ottawa', 'Pays-Bas': 'La Haye'}
{'France': 'Paris', 'Canada': 'Ottawa', 'Pays-Bas': 'La Haye'}
```

Tout comme le type `list`, ce type sert à *structurer les données*. L'exemple précédent fait correspondre des capitales à des pays.

7 Le type bool

Il sert à représenter les *valeurs booléennes*, valant `True` (vrai) ou `False` (faux).

Python

```
>>> False
False

>>> True
True
```

Cela peut paraître un peu pauvre, c'est trompeur : les *expressions logiques* sont des écritures dont la valeur est un booléen. Lorsque PYTHON les rencontre, il les évalue pour trouver soit `True` soit `False`.

Python

```
>>> 3 >= 2 # évalue si 3 est supérieur ou égal à 2
True

>>> 3 + 5 == 2 # évalue si 3 + 5 vaut 2
False

>>> 1 in [3, 4, 1, 5] # évalue si 1 est un élément de la liste
True
```

Remarque

> Pour *tester* si deux valeurs sont égales, on utilise == (et pas =).

Ce type dispose d'*opérations logiques* : `or` (ou), `and` (et) et `not` (non).

Python

```
>>> True and False # vrai que si les 2 sont vrais
False

>>> True or False # faux que si les 2 sont faux
True

>>> not (3 < 1) # contraire
True

>>> (2 < 1) or (3 >= 0)
True
```

Chapitre 11

Variables et affectations

1 Le symbole =

En mathématiques, le symbole $=$ a plusieurs significations

- dans $2+2=4$, on peut comprendre $=$ comme un opérateur d'évaluation : $2+2$, cela « donne » 4 ;
- dans $\mathcal{P}=2\times(\ell+\mathrm{L})$, on peut considérer que $=$ sert à définir ce qu'est le périmètre d'un rectangle de dimensions ℓ et L ;
- dans $3x+2=4x+5$, le $=$ sert à convenir que les 2 membres ont la même valeur et on cherche s'il existe un ou des nombres x qui satisfont l'égalité (appelée équation) ;
- *et cætera*.

En PYTHON, le symbole = n'a qu'un seul sens : il sert à l'*affectation*.

2 L'affectation

Il s'agit de « stocker » une valeur dans un endroit de la mémoire auquel PYTHON donne un nom[1]. Voici un exemple d'affectation :

```
a = 2
```

- 2 est une *valeur* de type `int` ;
- la *variable* a est créée ;
- a est « attachée » à la valeur 2 ;
- par extension a est également de type `int`.

Au cours d'un programme la valeur associée à une variable peut change...D'où le nom de *variable*.

Définition : affectation

Lors d'une affectation

- d'abord PYTHON évalue ce qu'il y a à droite du symbole = ;
- ensuite il affecte cette valeur à la variable qui figure à gauche du symbole = ;
- si la variable n'existe pas déjà, elle est créée automatiquement ;
- le type de la variable, c'est le type de la valeur qu'on lui affecte.

Que fait le programme suivant ?

[1] En réalité c'est plus compliqué mais cela ne nous intéresse pas.

Python

```
x = 0
x = x + 1
print(x)
```

- il crée une variable `x` de type `int` valant `0` ;
- il évalue `x + 1`, trouve `1` et affecte cette valeur à `x` ;
- évalue `x`, trouve 1 et donc affiche `1`.

À retenir

En mathématiques, $x = x + 1$ est une équation sans solution.

En PYTHON, l'instruction `x = x + 1` sert à augmenter la valeur de `x` de 1 (on dit aussi *incrémenter*).

2.1 Affectations multiples

PYTHON permet d'affecter plusieurs valeurs à plusieurs variables en même temps.

Python

```
>>> a, b = 10, 2
>>> a
10

>>> b
2

>>> a, b = b, a # permet d'échanger a et b
>>> a
2

>>> b
10
```

2.2 Notation condensée

On est souvent amené à écrire des instructions telles que `a = a + 1` ou `b = b / 2`. Cela peut être lourd quand les variables ne s'appellent pas `a` ou `b` mais `rayon_sphere` ou `largeur_niveau`. On peut utiliser les notation suivantes :

Python

```
>>> rayon_sphere = 3.4
>>> rayon_sphere /= 2
>>> rayon_sphere
1.7
```

```
>>> largeur_niveau = 19
>>> largeur_niveau += 1
>>> largeur_niveau
20
```

On dispose également de `*=`, `//=`, `%=`, `-=` et `**=`.

3 Le cas des variables de type str ou list

3.1 Les str

Les valeurs de type `str` sont composées de caractères *alphanumériques*. On peut accéder à chacun d'eux de la manière suivante :

Python

```
>>> chaine = 'Bonjour !'
>>> chaine[0]
'B'
>>> chaine[5]
'u'
```

Voici comment PYTHON représente la chaîne précédente :

chaine[i]	B	o	n	j	o	u	r		!

On a parfois besoin de connaître la longueur (*length* en anglais) d'une chaîne de caractères :

Python

```
>>> chaine = 'onzelettres'
>>> len(chaine)
11
```

On peut aussi accéder facilement au dernier (ou à l'avant dernier) caractère d'une variable de type `str` :

Python

```
>>> a = "M'enfin ?!"
>>> a[-1]
'!'

>>> a[-2]
'?'
```

3.2 Les list

Cela se passe un peu comme pour les `str`

Python

```
>>> lst = [3, 4, 8]
>>> lst[1] # élément d'indice 1 de lst
4

>>> lst[-1] # dernier élément de lst
8

>>> len(lst)
3
```

Lorsqu'on essaie d'accéder à un élément dont l'indice est supérieur ou égal à la longueur de la liste, on obtient une erreur. Dans l'exemple précédent si on évalue `lst[3]` on obtient :

```
IndexError: list index out of range
```

Une étude détaillée des `list` est faite au chapitre 15.

Chapitre 12

Tests et conditions

« Ceci n'est pas un test ! »

1 Des outils pour comparer

Ce sont les *opérateurs de comparaison* :

Opérateur	Signification	Remarques
<	strictement inférieur	Ordre usuel sur `int` et `float`, lexicographique sur `str`…
<=	inférieur ou égal	Idem
>	strictement supérieur	Idem
>=	supérieur ou égal	Idem
==	égal	« avoir même valeur » *Attention :* deux signes =
!=	différent	
`is`	identique	être le même objet
`is not`	non identique	
`in`	appartient à	avec `str`, `list` et `dict`
`not in`	n'appartient pas à	avec `str` et `list` et `dict`

Python

```
>>> a = 2
>>> a == 2
>>> a == 3
>>> a == 2.0
>>> a is 2.0
>>> a != 100
>>> a > 2
>>> a >= 2
```

Python

```
>>> a = 'Alice'
>>> b = 'Bob'
>>> a < b
>>> 'e' in a
>>> 'e' in b
>>> liste = [1,10,100]
>>> 2 in liste
```

Ces opérateurs permettent de réaliser des tests basiques. Pour des tests plus évolués on utilisera des « mots de liaison » logiques.

2 Les connecteurs logiques

- `and` permet de vérifier que 2 conditions sont *vérifiées simultanément*.
- `or` permet de vérifier qu'*au moins une* des deux conditions est vérifiée.
- `not` est un opérateur de *négation* très utile quand on veut par exemple vérifier qu'une condition est fausse.

Voici les tables de vérité des deux premiers connecteurs :

and	True	False
True	True	False
False	False	False

or	True	False
True	False	True
False	True	True

À ceci on peut ajouter que `not True` vaut `False` et vice-versa.

Python

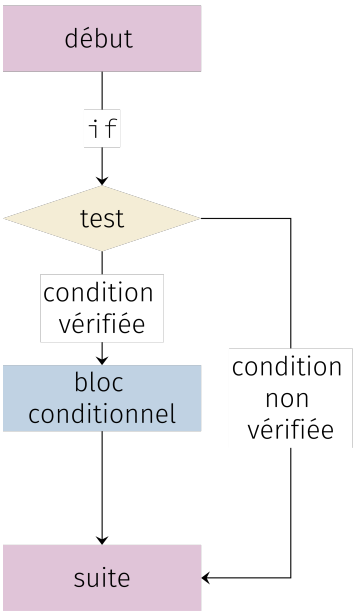
```
>>> True and False
>>> True or False
>>> not True
```

Python

```
>>> resultats = 12.8
>>> mention_bien = resultats >= 14 and resultats < 16
>>> print(mention_bien)
```

3 if, else et elif

Voici le schéma de fonctionnement d'un test `if` :

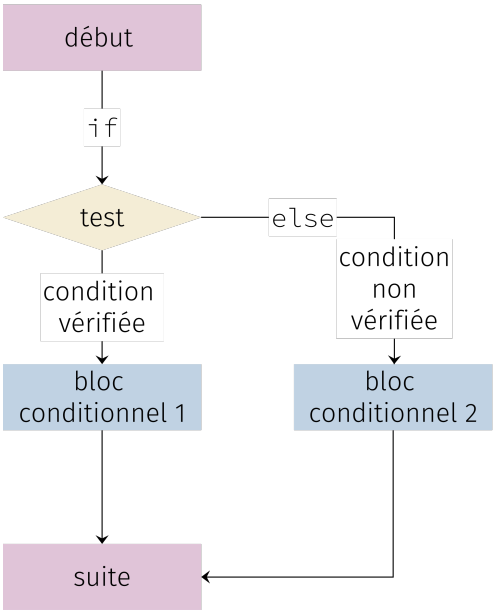


Attention : Un bloc conditionnel doit être *tabulé* par rapport à la ligne précédente : il n'y a ni `DébutSi` ni `FinSi` en PYTHON, ce sont les tabulations qui délimitent les blocs.

Python

```
phrase ='Je vous trouve très joli'
reponse = input('Etes vous une femme ?(O/N) : ')
if reponse == 'O':
    phrase += 'e'
phrase +='.'
print(phrase)
```

Voici le schéma de fonctionnement d'un test `if...else` :



Python

```
print('Bonjour')
age = int(input('Entrez votre age : '))
if age >= 18:
    print('Vous etes majeur')
else:
    print('Vous etes mineur.')
print('Au revoir.')
```

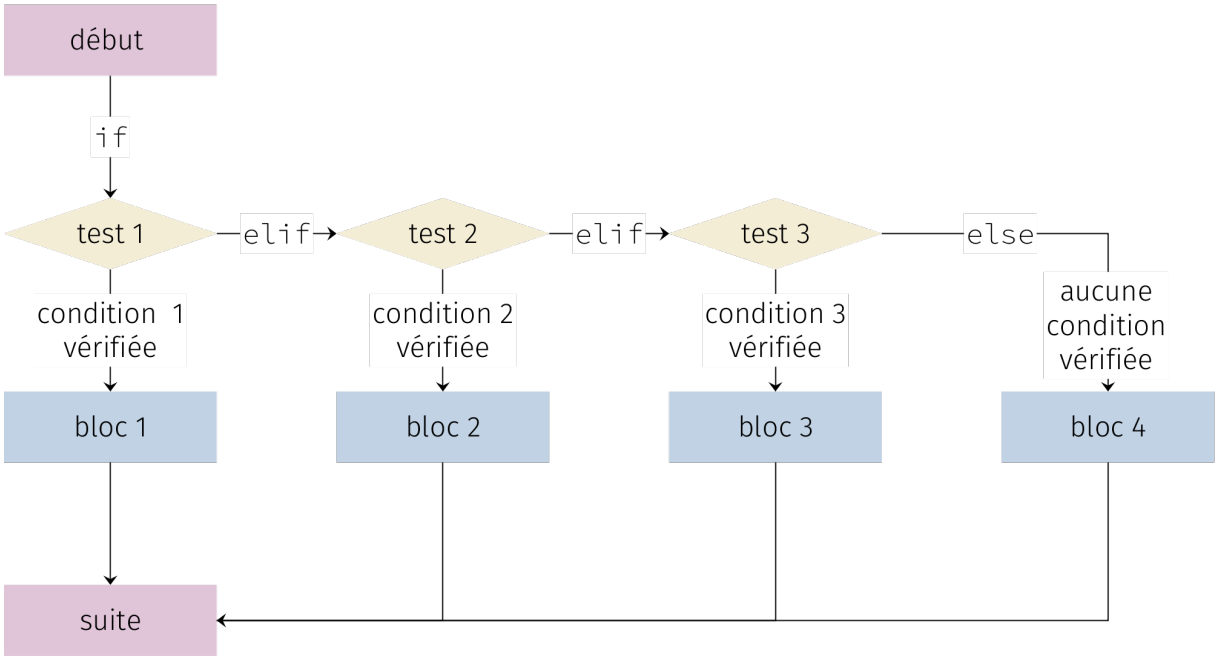
Voici un exemple de fonctionnement d'un test `if...elif...` :

Python

```
print('Bonjour')
prenom = input('Entrez un prénom : ')
if prenom == 'Robert':
    print("Robert, c'est le prénom de mon grand-père.")
elif prenom == 'Raoul':
    print("Mon oncle s'appelle Raoul.")
elif prenom == 'Médor':
    print("Médor, comme mon chien !")
```

```
else:
    print("Connais pas")
print('Au revoir.')
```

Et voici un schéma décrivant son fonctionnement :



On peut bien sûr inclure autant de `elif` que nécessaire.

4 Exercices

Exercice 131

Écrire un script qui demande son âge à l'utilisateur puis qui affiche `'Bravo pour votre longévité.'` si celui-ci est supérieur à 90.

Exercice 132

Écrire un script qui demande un nombre à l'utilisateur puis affiche si ce nombre est pair ou impair.

Exercice 133

Écrire un script qui demande l'âge d'un enfant à l'utilisateur puis qui l'informe ensuite de sa catégorie :

- trop petit avant 6 ans ;
- poussin de 6 à 7 ans inclus ;
- pupille de 8 à 9 ans inclus ;
- minime de 10 à 11 ans inclus ;
- cadet à 12 ans et plus ;

Exercice 134

Écrire un script qui demande une note sur 20 à l'utilisateur puis vérifie qu'elle est bien comprise entre 0 et 20. Si c'est le cas rien ne se produit mais sinon le programme devra afficher un message tel que `'Note non valide.'`.

Exercice 135

Écrire un script qui demande un nombre à l'utilisateur puis affiche s'il est divisible par 5, par 7 par aucun ou par les deux de ces deux nombres.

Exercice 136

En reprenant l'exercice du chapitre 1 sur les numéros de sécurité sociale, écrire un script qui demande à un utilisateur son numéro de sécurité sociale, puis qui vérifie si la clé est valide ou non.

Exercice 137

Écrire un script qui résout dans **R** l'équation du second degré $ax^2 + bx + c = 0$.
On commencera par `from math import sqrt` pour utiliser la fonction `sqrt`, qui calcule la racine carrée d'un `float`.
On rappelle que lorsqu'on considère une équation du type $ax^2 + bx + c = 0$

- si $a = 0$ ce n'est pas une équation de seconde degré ;
- sinon on calcule $\Delta = b^2 - 4ac$ et
  - Si $\Delta < 0$ l'équation n'a pas de solutions dans **R** ;
  - Si $\Delta = 0$ l'équation admet pour unique solution $\frac{-b}{2a}$;
  - Si $\Delta > 0$ l'équation admet 2 solutions : $\frac{-b-\sqrt{\Delta}}{2a}$ et $\frac{-b+\sqrt{\Delta}}{2a}$.

Pour vérifier que le script fonctionne bien on pourra tester les équations suivantes :

- $2x^2 + x + 7 = 0$ (pas de solution dans **R**) ;
- $9x^2 - 6x + 1 = 0$ (une seule solution qui est $\frac{1}{3}$) ;
- $x^2 - 3x + 2$ (deux solutions qui sont 1 et 2).

Exercice 138

L'opérateur **nand** est défini de la manière suivante : si **A** et **B** sont deux booléens alors

`A nand B` vaut `not (A and B)`

Construire la table de vérité de **nand** en complétant :

A	B	A and B	not (A and B)
False	False		
False	True		
True	False		
True	True		

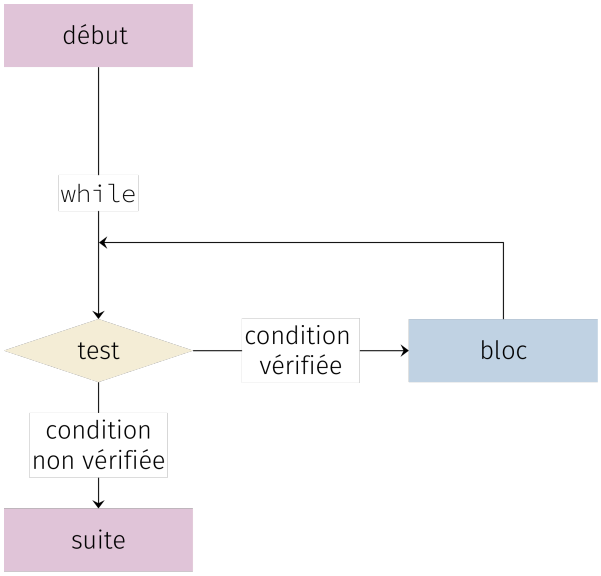
Chapitre 13
Boucles

« Tant que tu n'y arrives pas recommence. »

On s'intéresse dans ce chapitre aux *structures itératives*, plus communément appelée *boucles*.

1 La boucle while

Voici son schéma de fonctionnement :



La boucle `while` exécute un bloc d'instructions conditionnel *tant que* une condition est vérifiée. Dès que la condition n'est plus vérifiée, le bloc conditionnel n'est plus exécuté.

Python

```
reponse=''
print('Bonjour !')
while reponse !='n':
    reponse = input('Voulez-vous continuer ? (o/n) : ')
print('Au revoir.')
```

La boucle `while` doit être utilisée avec soin : si la condition est toujours vérifiée, le programme ne s'arrêtera pas :

Python

```
while True:
    print('Au secours !')
```

Voici un exemple typique d'utilisation de la boucle `while` :

On place un capital de 2000 euros sur un compte à intérêts annuels de 2%. On aimerait savoir au bout de combien de temps, sans rien toucher, le solde du compte dépassera 2300 euros.

Python

```
solde = 2000  # solde initial
n = 0  # nombre d'annees
while solde <= 2300:  # condition de boucle
    n += 1  # augmente le compteur d'annees
    solde *= 1.02  # actualise le solde
print(' Il nous faudra ', n, 'ans.')  # affichage final
```

2 La boucle for ... in range(...)

Commençons par examiner un nouveau type : `range` (plage de valeurs)

Python

```
>>> a = range(10)
>>> type(a)
>>> print(list(a))
```

Si `range(10)` ressemble beaucoup à la liste `[0,1,...,9]`, la finalité de `range(10)` est d'être un *itérateur*, c'est-à-dire une objet dont on peut parcourir le contenu pour créer une boucle :

Python

```
for i in range(10):
    print(i)
```

La syntaxe complète de `range` est : `range(<debut>,fin,<increment>)`.

Par défaut, si ce n'est pas précisé, `debut=0`, et `increment=1`.

`range(<debut>,fin,<increment>)` renvoie la plage de valeurs suivantes :

- On part de la valeur de début, appelons la `val`
- Tant que `val < fin` :
  - ajouter `val` à la plage
  - ajouter `increment` à `val`

Ainsi, `range(2,52,10)` renvoie la plage de valeurs `2,12,22,32,42`, mais `range(2,53,10)` renvoie la plage de valeurs `2,12,22,32,42,52`.

Très souvent, on se contente d'utiliser une instruction du type `range(n)`, où `n` est de type `int`.

Voici un exemple : Calculons $1+2+...+100$:

Python

```
somme = 0
for i in range(1, 101):
    somme += i
print(somme)
```

3 La boucle for ... in ...

On peut généraliser le paragraphe précédent à toute *variable itérable*, c'est extrêmement puissant : les `str`, les `list` et les `dict` sont des types itérables.

Voici des exemples :

Comptons le nombre de voyelles d'une chaîne de caractères :

Python

```
voyelles = 'aeiouy'  # ensemble de voyelles
phrase = input('Entrez une phrases sans accents : ').lower()  # phrase
 ↪  mise en minuscules
compteur = 0  # comptera les voyelles
for lettre in phrase:  # on parcourt la phrase
    if lettre in voyelles:  # est-ce une voyelle ?
        compteur += 1  # si oui on comptabilise
print('Nombre de voyelles : ', compteur)  # affiche le nombre
```

Faisons la moyenne d'une liste de notes :

Python

```
liste_notes = [12, 11.5, 13, 18, 13, 11, 9]
moyenne = 0
for note in liste_notes:
    moyenne += note
moyenne /= len(liste_notes)
print(moyenne)
```

Pour le dernier exemple on utilise le type `dict` : soit `a` une variable de ce type :

- `a.keys()` renvoie la liste des clés (des indices du dictionnaire).
- `a.values()` renvoie la liste des valeurs prises par le dictionnaire.

Voici un second programme de moyenne :

Python

```
resultats = {'EPS': 12, 'maths': 15, 'info': 18}
moyenne = 0
for note in resultats.values():
    moyenne += note
moyenne /= len(resultats)
```

```
print(moyenne)
```

4 Quelle boucle utiliser ?

Si la boucle dépend d'une condition particulière on préfèrera la boucle `while`.
Si le nombre d'itérations de la boucle est connu on préfèrera une boucle `for`.
On peut utiliser une boucle `for` sur toute *structure itérative*, par exemple une variable de type `range`, `str`, `list` ou, dans une certaine mesure, `dict`.

5 Exercices

Exercice 139

Calculer à l'aide d'un script la somme des carrés des 1000 premiers entiers non nuls.

Exercice 140

Calculer à l'aide d'un script la somme des carrés des 1000 premiers multiples de 3 non nuls.

Exercice 141

Écrire un script qui demande une phrase à l'utilisateur, puis affiche la phrase en rajoutant des tirets.
Exemple : on entre `'Salut à toi'` le script affiche `'S-a-l-u-t- -à- -t-o-i-`.

Exercice 142

Calculer à l'aide d'un script le nombre n à partir duquel la somme $1^2 + 2^2 + \ldots + n^2$ dépasse un milliard.

Exercice 143

Écrire un script qui demande une phrase et compte le nombre d'occurrences de la lettre « a » dans celle-ci.

Exercice 144

Programmer le jeu du "plus petit plus grand" :

- L'ordinateur choisit un nombre entier au hasard compris entre 0 et 100.
  Au début du script, importer la fonction `randint` du module `random` avec `form random import randint`.
  Pour obtenir un entier au hasard, utiliser `randint(0,100)`.
- L'utilisateur propose un nombre, l'ordinateur répond « gagné » , « plus petit » ou « plus grand » .
- Le programme continue tant que l'utilisateur n'a pas gagné.

Exercice 145

On considère la suite s définie par :

$$\begin{cases} s_0 &= 1000 \\ s_{n+1} &= 0,99s_n + 1 \quad \text{pour tout } n \in \mathbf{N} \end{cases}$$

- Écrire un script calculant les premiers termes de s (vous décidez le nombre de termes).
- Utiliser ce script pour conjecturer la limite de s.
- Modifier ce script pour obtenir le plus petit entier n tel que l'écart entre s_n et sa limite soit inférieur ou égal à 10^{-4}.

Exercice 146

φ (lettre phi, équivalent du « f » en grec) est défini par :

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ddots}}}$$

Sur papier, fabriquer une suite par récurrence commençant ainsi :

- 1
- $1 + \frac{1}{1}$
- $1 + \cfrac{1}{1 + \frac{1}{1}}$
- Et cætera (trouver une relation simple pour calculer le terme suivant à partir du terme actuel).

Programmer un script qui calcule successivement les termes de cette suite (aller jusqu'à 10000^{e}).

Comparer avec la valeur exacte de φ, qui est $\frac{1+\sqrt{5}}{2}$.

Exercice 147

Écrire un script qui détermine si un entier est premier ou pas.

Chapitre 14
Fonctions

« Quelle est la fonction de ce chapitre ? »

1 Exemples de fonctions

1.1 Un objet déjà connu

Nous avons déjà rencontré des fonctions *côté utilisateur* :

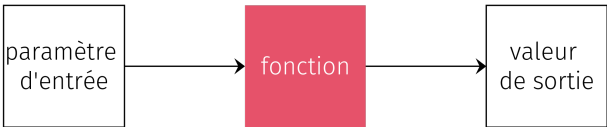
- `input`
  - prend en entrée une chaîne de caractères ;
  - renvoie la chaîne de caractère saisie par l'utilisateur.

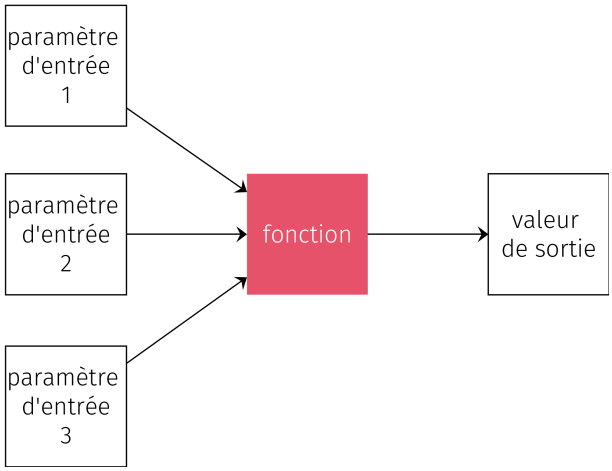
  On peut noter ceci `input(chaine: str) -> str`
- `len`
  - prend en entrée une liste ;
  - renvoie le nombre d'éléments de cette liste.

  On peut noter cela `len(lst: list) -> int`

1.2 De multiples formes

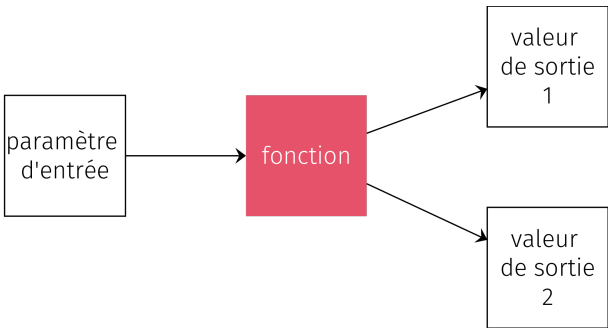
Les deux exemples précédents rentrent dans la catégorie représentée à droite.

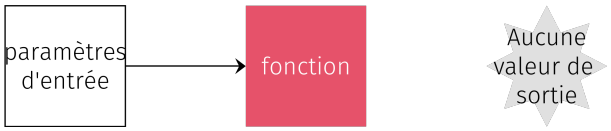




Certaines fonctions sont comme à gauche. Par exemple `max(20,3,10)` renvoie 20.

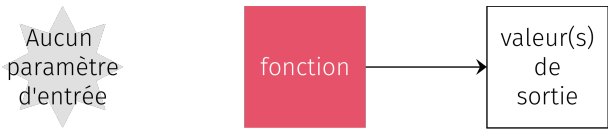
D'autres fonctions sont comme à droite.
On verra des exemples plus tard.





D'autres encore sont comme à gauche.
Par exemple `print("salut")` ne renvoie rien mais affiche `salut` à l'écran.

D'autres suivent le schéma ci-contre.
Par exemple dans le module `time`, la fonction `time` ne prend aucun paramètre d'entrée mais renvoie l'heure qu'indique l'horloge de l'ordinateur.
On peut par exemple l'utiliser pour stocker une heure précise en tapant `maintenant = time()`.



Enfin certaines suivent ce schéma.
Par exemple dans le module `pygame`, `pygame.display.flip` ne prend aucun paramètre d'entrée, ne renvoie aucune valeur, mais actualise la fenêtre graphique.
On l'appelle donc en tapant `pygame.display.flip()`.

Il est possible de créer de nouvelles fonctions.
On parle alors de fonctions *côté concepteur*.

Il faut donc définir rigoureusement ce qu'est une fonction.

2 Définition de la notion de fonction

Définition : fonction

Une *fonction* est un « morceau de code » qui représente un *sous-programme*.
Elle a pour but d'effectuer une tâche *de manière indépendante*.

Exemple

On veut modéliser la fonction mathématique f définie pour tout nombre réel x par

$$f(x) = x^2 + 3x + 2$$

On écrira alors

```
def f(x : float) -> float:
    return x ** 2 + 3 * x + 2
```

Pour évaluer ce que vaut $f(10)$ et affecter cette valeur à une variable, on pourra désormais écrire `resultat = f(10)`.

Que fait la fonction `mystere` ?

```
def mystere(a : float, b : float) -> float:
    if a <= b:
        return b
    else:
        return a
```

La fonction `mystere` :

- prend en entrée deux paramètres de type `float` `a` et `b` ;
- renvoie le plus grand de ces deux nombres.

La réponse que l'on vient de formuler s'appelle *la spécification* de la fonction *f*.

Définition : fonction

Donner la spécification d'une fonction `f` c'est

- préciser le(s) type(s) du (des) paramètre(s) d'entrée (s'il y en a) ;
- indiquer sommairement ce que fait la fonction `f` ;
- préciser le(s) type(s) de la (des) valeur(s) de sortie (s'il y en a).

3 Anatomie d'une fonction

Python

```
def f(lst: list) -> int
    mini = lst[0]
    n = len(lst)
    for i in range(n):
        lst[i] < mini:
        mini = lst[i]
    return mini
```

La fonction `f`

- prend en entrée une liste (sous entendu d'entiers) ;
- renvoie le plus petit entier de cette liste.

3.1 Paramètre formel

```
def f(lst: list) -> int
    mini = lst[0]
    n = len(lst)
    for i in range(n):
        lst[i] < mini:
        mini = lst[i]
    return mini
```

paramètre formel

Le paramètre d'entrée est *formel* : *le nom de cette variable n'existe qu'à l'intérieur de la fonction*. Si ce nom de variable existe déjà à l'extérieur de la fonction, *ce n'est pas la même variable*.

Le type du paramètre d'entrée peut être spécifié. Ce n'est pas obligatoire mais très fortement recommandé pour « garder les idées claires » .

```
def f(lst: list) -> int
    mini = lst[0]
    n = len(lst)
    for i in range(n):
        lst[i] < mini:
        mini = lst[i]
    return mini
```

type du paramètre formel

3.2 Variables locales

```
def f(lst: list) -> int
    mini = lst[0]
    n = len(lst)
    for i in range(n):
        lst[i] < mini:
        mini = lst[i]
    return mini
```

variables locales

Toutes les variables *créées* dans une fonction n'existent *que dans cette fonction*. Elles ne sont pas accessibles depuis l'extérieur de la fonction. On dit que ce sont des *variables locales*.

3.3 Valeur de sortie

Le type de la valeur de sortie peut être précisé, c'est également recommandé.

```
def f(lst: list) -> int
    mini = lst[0]
    n = len(lst)
    for i in range(n):
        lst[i] < mini:
        mini = lst[i]
    return mini
```

type de la valeur renvoyée

4 En pratique

4.1 Des exemples

Python

```
1   def f(x : float) -> float:
2       return x ** 2 + 3 * x + 2
3
4   print(f(1)) # Affiche 6
```

Le programme commence à la ligne 4 !
Les 2 premières lignes servent à définir la fonction `f`, elles ne sont exécutées que lorsqu'on évalue `f(1)`.

Python

```
def f(x : float) -> float:
    return x ** 2 + 3 * x + 2

print(x) # Provoque une erreur
```

L'erreur vient du fait que la variable `x` *n'est pas définie*. Le « `x` qu'on voit dans la fonction `f` » est un paramètre formel et n'existe que dans `f`.

Python

```
def f(x : float) -> float:
    a = 2
    return x + a

print(a) # Provoque une erreur
```

L'erreur vient du fait que la variable `a` *est locale* : elle n'est définie que durant l'exécution de `f`.

Python

```
def f(x : float) -> float:
    a = 2
    return x + a

print(f(4)) # Affiche 6
print(a) # Provoque une erreur
```

C'est encore la même erreur : une fois `f(4)` évaluée, `a` n'existe plus.

Python

```
1   def f(x : float) -> float:
2       a = 2
3       return x + a
```

```
4
5   a = 3
6   print(f(4)) # Affiche 6
7   print(a) # Affiche 3 et pas 2
```

La variable `a` définie dans la fonction `f` n'est pas la même que celle qui est définie à la ligne 5.
Celle définie à la ligne 2 est *locale*.
La variable `a` de la ligne 5 est appelée *globale*.

Python

```
def f(x : float) -> float:
    return x + a

a = 3
print(f(4)) # Affiche 7
```

À retenir

Une fonction a le droit d'*accéder en lecture* à une variable globale, mais n'a pas *a priori* le droit d'en modifier la valeur.

4.2 À éviter autant que possible

Python

```
1   def f(x : float) -> float:
2       global a
3       a = a + 1
4       return x + a
5
6   a = 3
7   print(f(4)) # Affiche 8
8   print(a) # Affiche 4
```

À la ligne 2, on signale à Python que `f` a la droit de modifier la variable globale `a`. C'est fortement déconseillé : sauf si on ne peut pas faire autrement, une fonction ne doit pas modifier les variables globales.

Chapitre 15
Listes

Le type `list` permet de stocker des valeurs dans un ordre précis.

Python

```
# on crée une liste avec 3 valeurs
lst = ['bonjour', 3.14, True]
```

Une valeur de type `list` est itérable :

- on peut accéder à un élément de la liste, par exemple `lst[0]` ;
- on peut parcourir une liste.

Pour accéder à un élement d'une liste situé à un endroit précis, on doit connaître son *indice* : le premier élément d'une liste a l'indice zéro, le deuxième l'indice 1, *et cætera*.

Python

```
# on crée une liste avec 3 valeurs
>>> lst = ['bonjour', 3.14, True]
# premier élément : indice 0
>>> lst[0]
'bonjour'

# deuxième élément
>>> lst[1]
3.14
```

1 Opérations de base

1.1 Créer une liste

- `lst = list()` crée une liste vide ;
- `lst = []` fait la même chose ;
- `lst = ['a', 7, True]` crée une liste composée de 3 éléments.

Une liste peut contenir des éléments de plusieurs types mais en pratique on évite cela.

1.2 Modifier un élément

Le type `list` est *mutable* : on peut changer un ou des éléments d'une liste *sans changer la liste elle-même*.

Python

```
>>> lst = [2, 3, 4, 1]
# on change le deuxième élément
>>> lst[1] = 10
>>> lst
[2, 10, 4, 1]
```

1.3 Ajouter un élément en fin de liste

On reprend l'exemple précédent

Python

```
>>> lst.append(7) # ajoute 7 à la fin de la liste
>>> lst
[2, 10, 4, 1, 7]
```

Remarque

`lst = lst + [7]` a le même effet que `lst.append(7)` : on crée une « mini-liste » `[4]`, on concatène les 2 listes et on remet le résultat dans `lst`.
En pratique la première méthode est la plus simple et aussi la plus rapide.

1.4 Insérer un élément à une place donnée

Pour une liste `lst` valant `[2, 10, 4, 1]`, si on veut insérer la valeur 5 à l'indice 1 on écrira :

```
lst.insert(1, 5)
```

et `lst` vaudra `[2, 5, 10, 4, 1]`

La syntaxe est `lst.insert(indice, valeur)`

1.5 Retirer un élément à une position donnée

Si une liste `lst` a pour valeur `[3 ,7, 1]` et qu'on veut supprimer son deuxième élément alors on écrit :

```
del lst[1]
```

Ensuite, `lst` aura la valeur `[3,1]`.

1.6 Retirer une valeur précise

Pour retirer une valeur *qui appartient à une liste* on procède ainsi :

Si `lst` a la valeur `[1, 2, 5, 4, 2, 3]` alors l'instruction

`lst.remove(2)`

Supprime la *première occurrence* de 2 dans `lst`.
Après cela, `lst` a la valeur `[1, 5, 4, 2, 3]`.

1.7 Concaténer des listes

On peut procéder de 2 manières :

- `lst1.extend(lst2)` ajoute les éléments de la liste `lst2` à la fin de `lst1`;
- `lst1 = lst1 + lst2` crée une liste avec les éléments de `lst1` et ceux de `lst2`, puis replace le résultat dans `lst1`.

En pratique la première méthode est plus rapide.

1.8 Longueur d'une liste

La fonction `len`

- prend en entrée une liste `lst`;
- renvoie la longueur de cette liste.

Ainsi `len([2, 3, 4])` vaut 3.

1.9 Divers

`lst.sort()` trie la liste dans l'ordre croissant.

`lst.reverse()` met les éléments dans l'ordre inverse.

2 Opérations avancées

2.1 Copier une liste (mauvaise méthode)

Python

```
>>> lst1 = [5, 6, 8]
>>> lst2 = lst1
>>> lst1[0] = 10
>>> lst1
[10, 6, 8]

>>> lst2
[10, 6, 8]  # problème : lst2[0] a changé aussi !
```

Ce comportement « étrange » vient du fait que le type `list` est *mutable*. Nous allons expliquer cela plus tard dans ce chapitre.

2.2 Copier une liste (bonne méthode)

Python

```
>>> lst1 = [5, 6, 8]
>>> lst2 = lst1[:] # on copie tous les éléments de lst1 dans lst2
>>> lst1[0] = 10
>>> lst1
[10, 6, 8]

>>> lst2
[5, 6, 8] # Ouf !
```

2.3 Extraire une sous-chaîne

Soit `lst` une liste de longueur n et p et q deux entiers tels que $0 \leqslant p < q \leqslant n$. Alors

- `lst[p:q]` est la liste composée des éléments `lst[p]`, ..., `lst[q-1]` ;
- `lst[:q]` signifie `lst[0:q]` ;
- `lst[p:]` signifie `lst[p:n]`.

Exemple

Si `lst` vaut `[2, 5, 3, 4, 9, 2, 5]` alors

- `lst[2:6]` vaut `[3, 4, 9, 2]` ;
- `lst[3:]` vaut `[4, 9, 2, 5]` ;
- `lst[:2]` vaut `[2, 5]`.

3 Parcourir une liste

3.1 Parcours selon les indices

Définition : parcours selon les indices

Soit `lst` une liste de longueur `n`.

Alors ses éléments sont `lst[0]`, ..., `lst[n-1]` et on parcourt la liste en

- considérant un entier `i` qui jouera le rôle d'*indice* ;
- faisant parcourir à `i` la plage de valeurs `range(n)` ;
- considérant les `lst[i]`.

Exemple : un parcours selon les indices

On affiche les élements d'une liste grâce à un parcours par les indices.

```
lst = [54, 65, 123]
```

```
n = len(lst) # n vaut 3
for i in range(n): # range(3), c'est 0, 1, 2
    print(lst[i])
```

Le parcours d'une liste par les indices est *crucial* lorsque lors du parcours, on veut savoir à quelle place on se trouve dans la liste. C'est le cas quand on veut déterminer si une liste est triée dans l'ordre croissant ou non : il faut regarder si chaque élement est plus petit que le suivant dans la liste.
C'est aussi le cas quand on veut par exemple déterminer l'indice de la première apparition d'une valeur dans une liste.

3.2 Parcours selon les éléments

C'est plus simple que le parcours selon les indices mais on pert un peu d'information car pendant le parcours, on ne sait pas à quelle place on se trouve dans la liste.

Définition : parcours selon les éléments

Le parcours des éléments d'une liste `lst` s'effectue à l'aide d'une simple boucle `for x in lst`.
`x` prend alors successivement les valeurs de chacun des éléments de `lst`, dans l'ordre.

Exemple : un parcours selon les éléments

```
lst = [54, 65, 123]
for x in lst:
    print(x)
```

3.3 Bilan

On peut toujours utiliser un parcours de liste selon les indices. On peut toujours transformer un parcours selon les éléments en un parcours selon les indices. Le contraire est faux si l'on a *absolument* besoin de savoir quels sont les indices des éléments que l'on examine lors du parcours.

À éviter absolument

Les codes comme celui-ci :

```
for i in lst:
    print(i)
```

On a l'impression que `i` est un indice mais c'est une valeur, et l'expérience prouve que dans 90% des cas, une erreur du type `lst[i]` survient, alors que...`i` n'est pas un indice ici !

De même les codes comme celui-là :

```
for x in range(len(lst)):
    print(lst[x])
```

Là encore il y a beaucoup de chances que l'indice `x` soit pris pour une valeur.

Conseil

Réserver les noms de variables `i`, `j` et `k` pour les indices et `x`, `y` et `z` pour les éléments.

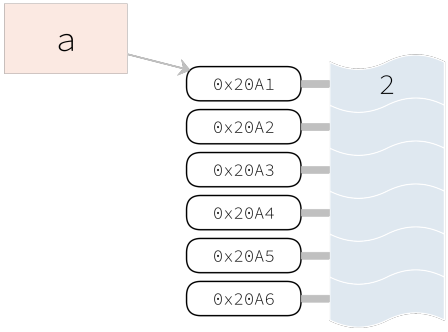
4 Mutabilité

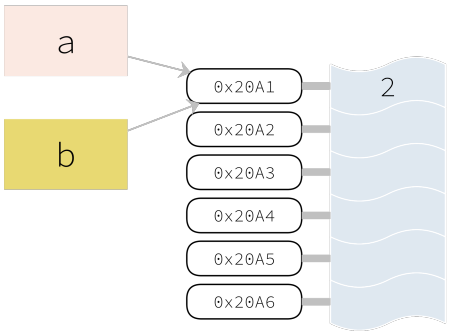
Examinons la différence entre un type *non mutable* tel que `int` et le type `list`, qui est *mutable*.

4.1 Variables de type non-mutable

```
a = 2
```

La valeur 2 est stockée en mémoire et une variable **a** est créée, associée à cette valeur.



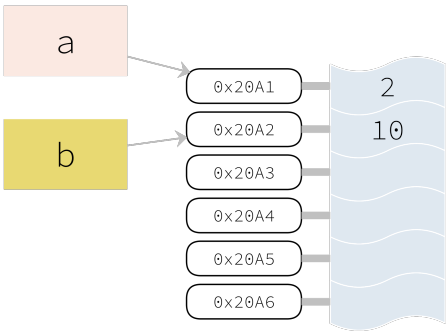



```
b = a
```

Une deuxième variable **b** est créée, avec pour valeur 2 également. Elles partagent la même adresse-mémoire.

```
b = 10
```

La valeur 10 est stockée dans une autre adresse mémoire (car la valeur 2 sert toujours pour **a**) et associée à **b**.

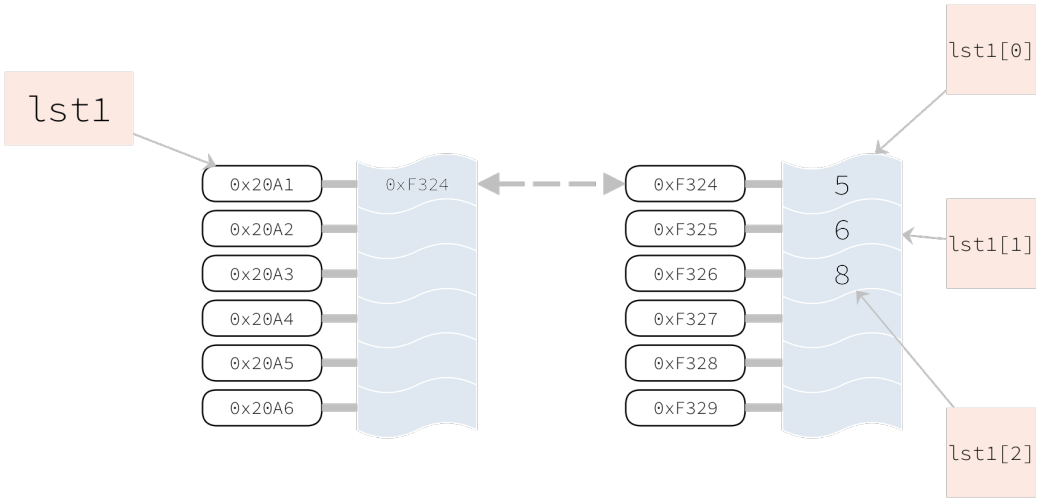


4.2 Variables de type mutable

Copier une liste (mauvaise méthode)

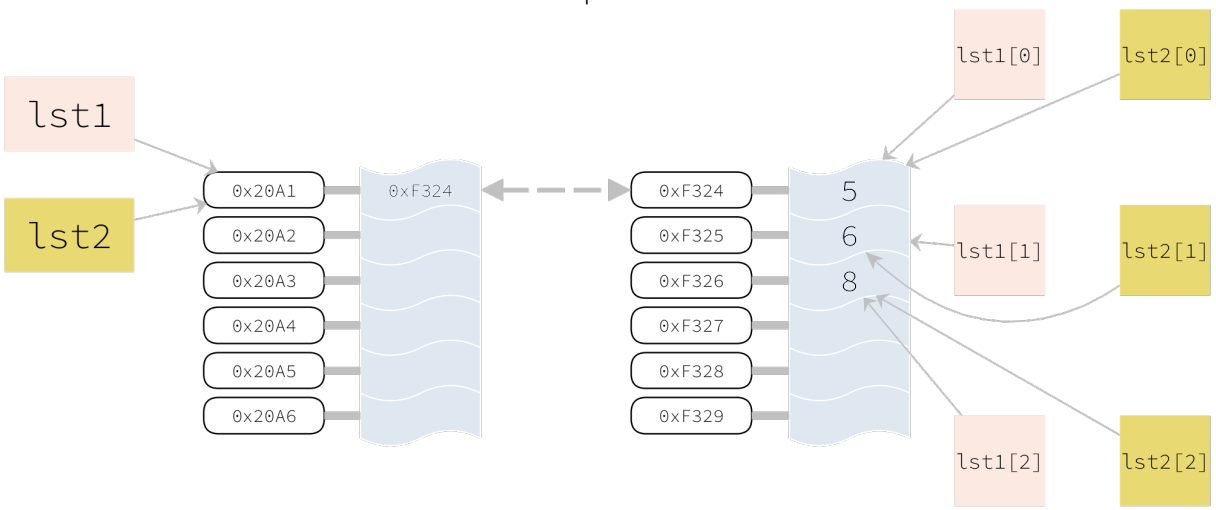
```
lst1 = [5, 6, 8]
```

Les élements 5, 6 et 8 sont stockés en mémoire et `lst1` contient l'adresse du début de la plage mémoire à laquelle ces valeurs sont stockées.



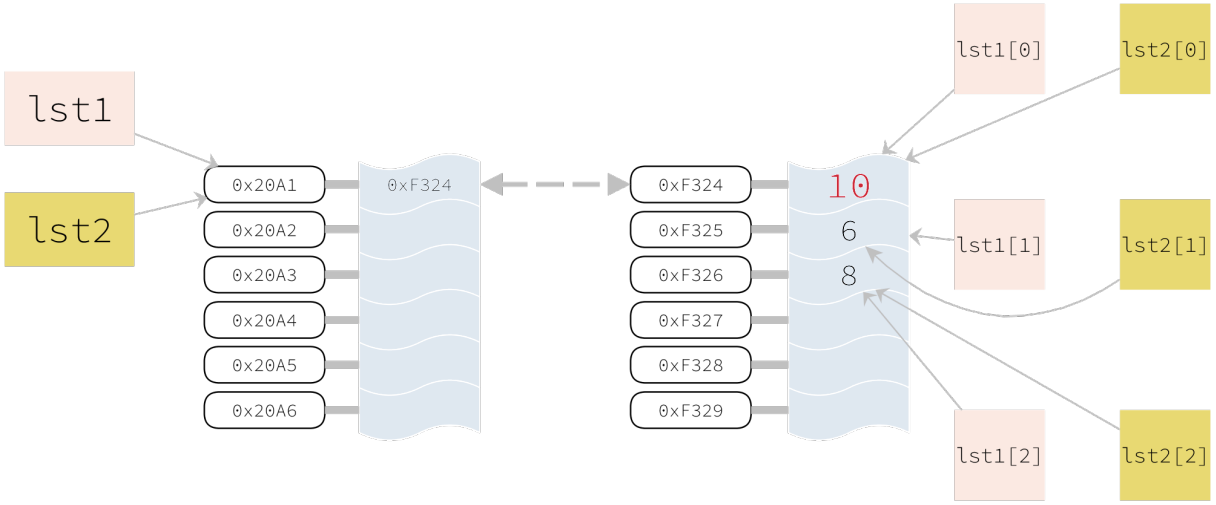
`lst2 = lst1`

La variable `lst2` est associée à la même valeur que `lst1`.



`lst1[0] = 10`

La valeur de `lst1[0]` est changée, elle l'est donc aussi pour `lst2`.

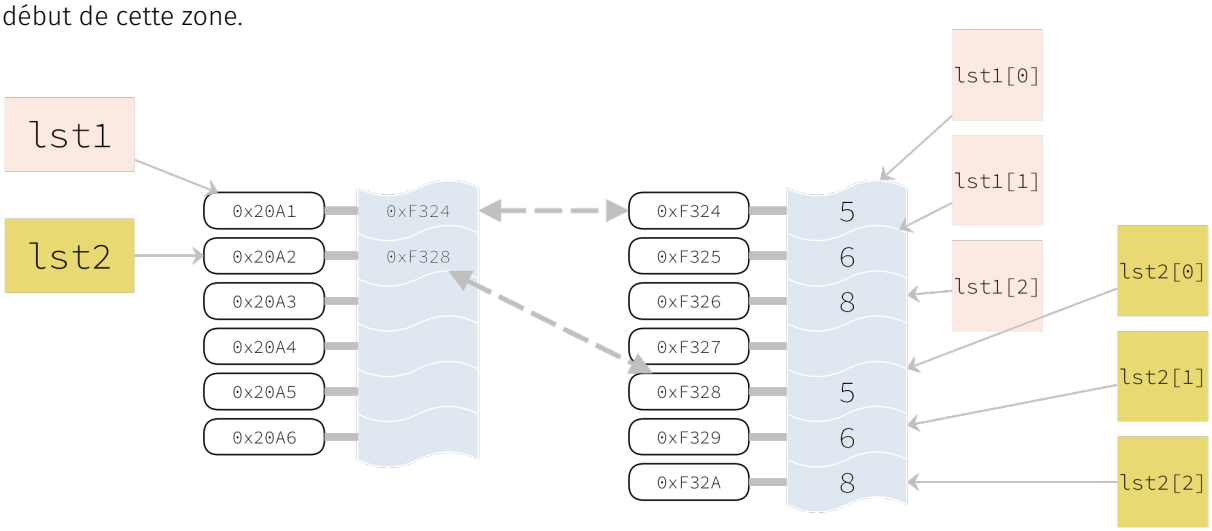


Voilà donc pourquoi lorsqu'on écrit `lst2 = lst1`, tout changement dans `lst1` se reflète aussi dans `lst2` et vice-versa.

Copier une liste (bonne méthode)

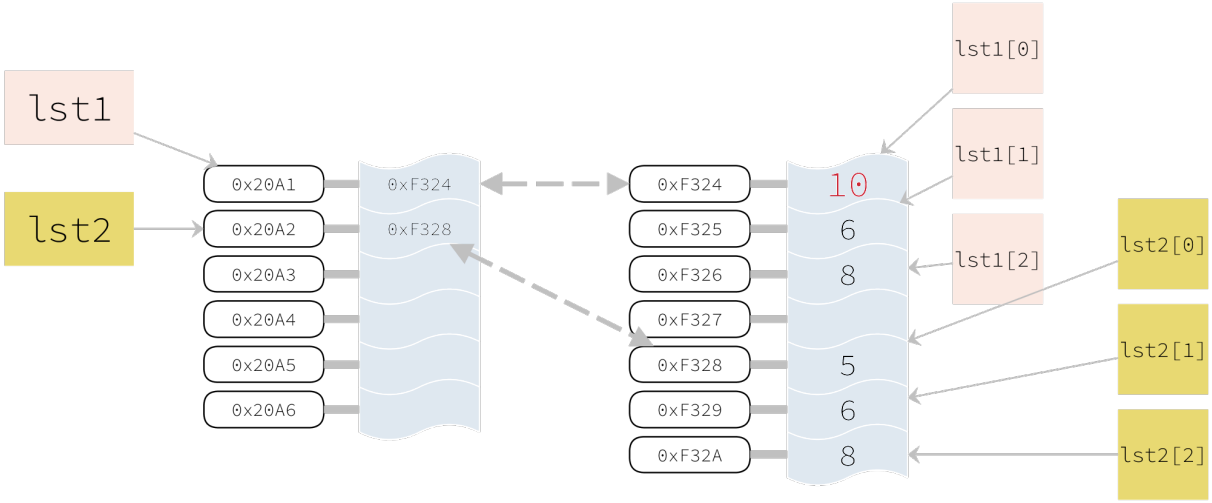
```
lst2 = lst1[:]
```

Les éléments de `lst1` sont recopiés dans une autre zone mémoire, et `lst2` « pointe » sur l'adresse du début de cette zone.




```
lst1[0] = 10
```

Le changement n'affecte pas `lst2`.



Chapitre 16

Écriture en compréhension

1 Écritures simples

Jusqu'à présent, pour construire des listes on a souvent :

- créé une liste `lst` vide ;
- construit une boucle `for` ou `while` ;
- peuplé la liste avec `lst.append`.

Exemple

```
lst = []
for i in range(10):
    lst.append(i*i)
print(lst)
```

Ce programme affiche `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
C'est la liste des carrés des 10 premiers entiers naturels.
En mathématiques, l'ensemble des carrés des 10 premiers entiers naturels se note

$$\{i^2 \mid i \in \mathbf{N},\, i < 10\}$$

C'est une écriture en *compréhension*.
On peut faire la même chose en PYTHON :

```
lst = [i*i for i in range(10)]
```

Évidemment, c'est plus rapide que la méthode précédente… Et on peut faire bien plus ! On peut utiliser une liste pour en construire une autre, par exemple en ajoutant 1 à chacun des éléments :

Python

```
>>> lst1 = [2, -1, 3, 4, 7]
>>> lst2 = [x + 1 for x in lst1]
>>> lst2
[3, 0, 4, 5, 8]
```

Dans le même esprit, on peut construire une liste dont les élements sont ceux de la première, mais avec une conversion de type :

Python

```
>>> lst1 = ['2', '0', '13']
>>> lst2 = [int(x) for x in lst1]
>>> lst2
[2, 0, 13]
```

Ou encore fabriquer la liste des initiales à partir d'une liste de prénoms :

Python

```
>>> lst1 = ['Fred', 'Titouan', 'Tinaïg']
>>> lst2 = [prenom[0] for prenom in lst1]
>>> lst2
['F', 'T', 'T']
```

2 Écritures avec conditions

Il est possible d'utiliser `if` en compréhension : mettons dans `lst2` le double de chaque élément de `lst1` supérieur à 10 (dans l'ordre de parcours).

Python

```
l>>> lst1 = [8, 0, 11, 10, 3, 15]
>>> lst2 = [2 * x for x in lst1 if x > 10]
>>> lst2
[22, 30]
```

Il est possible d'utiliser `if ... else ...` en compréhension, mais à ce moment là il faut écrire les conditions au début : créons une nouvelle liste en remplaçant tous les nombres négatifs de `lst1` par zéro.

Python

```
>>> lst1 = [8, -10, 11, -4, -3, 15]
>>> lst2 = [(x if x > 0 else 0) for x in lst1]
# les parenthèses sont facultatives
>>> lst2
[8, 0, 11, 0, 0, 15]
```

Créons une liste contenant les indices des éléments de `lst1` qui sont strictement positifs.

Python

```
>>> lst1 = [8, -10, 11, -4, -3, 15]
>>> lst2 = [i for i in range(len(lst1)) if lst1[i] > 0]
>>> lst2
[0, 2, 5]
```

3 Les écritures en compréhension imbriquées

0	0	0	0
0	0	0	0
0	0	0	0

Si on veut représenter ce « tableau de nombres » par une liste, on peut écrire cet liste de 3 lignes comportant chacune 4 éléments :

`lst = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]`.

Cependant il est plus pratique d'écrire

```
lst=[[0 for j in range(4)] for i in range(3)]
```

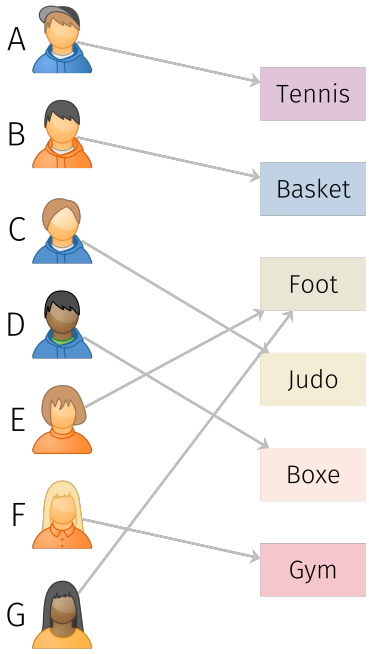
4 Pour conclure

On peut combiner toutes les techniques que nous venons de voir. Par exemple on peut créer une liste de listes de listes avec des conditions, *et cætera*. La seule limite, c'est l'imagination et la capacité à écrire en PYTHON !

Chapitre 17

Dictionnaires

1 Un nouveau type



On demande à des jeunes quel est leur sport préféré, les résultats sont présentés sur la figure ci-contre.
Un sport peut être cité par *plusieurs* jeunes, en revanche chaque jeune ne peut citer qu'*un seul* sport.
On pourrait utiliser une ou plusieurs listes pour représenter ces données mais il y a mieux : le *dictionnaire*. La variable `sport` est de type `dict` :

Python

```
sport = {'A': 'Tennis', 'B': 'Basket',
         'C': 'Judo', ' D': 'Boxe',
         'E': 'Foot', 'F': 'Gym',
         'G': 'Foot'}
```

Définition : dictionnaire

Un dictionnaire est un ensemble d'*éléments*.
les éléments sont des couples de la forme *clé : valeur*.

La syntaxe est :
`variable = { cle1 : valeur1, cle2 : valeur2, ...}`

Les valeurs peuvent être de n'importe quel type. Les clés peuvent être

- des `bool`, des `int`, des `float` ;
- des `str`…
- mais pas des `list` !

On peut tout de même utiliser des `tuples` en guise de clés : les `tuples` ressemblent aux `list` mais sont *non mutables*.

`a = (1, 2, 3)` est un exemple de `tuple`.

2 Opérations sur les dictionnaires

2.1 Accéder à une valeur par sa clé

Pour connaître le sport préféré de `'A'`, c'est simple :

Python

```
>>> sport['A']
'Tennis'
```

2.2 Créer de nouveaux couples clé : valeur

Contrairement aux listes, il n'y a pas de méthode `append`.
Pour intégrer l'information « le sport préféré de H est le Rugby » on écrira simplement :

Python

```
>>> sport['H']='Rugby'
```

2.3 Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide et remplir ses valeurs au fur et à mesure :

Python

```
>>> d = dict()
>>> d['bonjour'] = 'hello'
>>> d['crayon'] = 'pencil'
>>> d['se prélasser'] = 'to bask'
```

2.4 Supprimer un élément du dictionnaire

`del d['crayon']` supprime l'élément `'crayon': 'pencil'`.

2.5 Fusionner 2 dictionnaires

```
>>> d1 = {"anglais": "bread",
          "français": "pain",
          "slovaque": "chlieb"}
>>> d2 = {"allemand": "brot", "italien": "pane"}
>>> d1.update(d2) # fusionne d2 dans d1
>>> d1
{"anglais": "bread", "français": "pain", "slovaque": "chlieb", "allemand":
 ↪  "brot", "italien": "pane"}
```

2.6 Parcourir l'ensemble des clés d'un dictionnaire

Python

```
for cle in d1.keys():
    print(cle)
```

Ce script affiche

```
anglais
français
slovaque
allemand
italien
```

2.7 Parcourir l'ensemble des valeurs d'un dictionnaire

Python

```
for valeur in d1.values():
    print(valeur)
```

Ce script affiche

```
bread
pain
chlieb
brot
pane
```

2.8 Précisions

`d1.keys()` et `d1.values()` ressemblent à des listes mais n'en sont pas![1]

Pour avoir par exemple la liste des clés de `d1` on écrira :

```
list(d1.keys())
```

ou bien en *compréhension* (ce qui revient au même mais peut s'avérer utile)

```
[k for k in d1.keys()]
```

2.9 Erreurs de clé

```
print(d1['suédois'])
```

Ce script produit une erreur :

```
KeyError : 'suédois'
```

[1] Ce sont des *itérateurs*, sctructures destinées à être parcourues.

Exemple : Utilisation d'un dictionnaire

On veut créer un tableau de 10 × 10 cases avec la valeur 0 dedans.

On peut bien sûr créer cela avec une liste de listes (en compréhension) mais on peut également utiliser un dictionnaire :

```
{(x, y) : 0 for x in range(1, 11) for y in range(1, 11)}
```

Avantages :

- plus simple à manipuler : on écrit `d[x, y]` au lieu de `d[x][y]` ;
- on n'est pas obligé de faire commencer les indices à zéro.

Inconvénients :

- prend plus de place en mémoire (on s'en fiche un peu) ;
- plus flexible entraîne plus de possibilité d'erreurs !

3 Utilisation des dictionnaires

Typiquement, pour stocker des *données structurées* :

```
reseau = {'nom'        : 'local',
          'ip'         : '192.168.1.0',
          'masque'     : '255.255.255.0',
          'passerelle' : '192.168.1.254'}
```

On utilise fréquemment des listes de dictionnaires, ou bien des dictionnaires de listes.

Table des matières