



# Table des matières

<b>1</b>	<b>Logique</b>	<b>5</b>
1.1	Propositions . . . . .	5
1.2	Équivalences logiques . . . . .	7
1.3	Fonctions propositionnelles . . . . .	8
1.4	Quantificateurs . . . . .	9
1.5	Applications . . . . .	10
<b>2</b>	<b>Ensembles</b>	<b>13</b>
2.1	Relations ensemblistes . . . . .	13
2.2	Opérations ensemblistes . . . . .	14
2.3	Autres opérations . . . . .	15
2.4	Opérations généralisées . . . . .	17
<b>3</b>	<b>Fonctions</b>	<b>19</b>
3.1	Définitions de base . . . . .	19
3.2	Fonctions usuelles . . . . .	20
3.3	Injectivité, surjectivité, bijectivité . . . . .	21
3.4	Composition de fonctions . . . . .	24
<b>4</b>	<b>Suites et sommes</b>	<b>27</b>
4.1	Suites numériques . . . . .	27
4.2	Suites récursives . . . . .	28
4.3	Mots . . . . .	29
4.4	Sommes . . . . .	30
<b>5</b>	<b>Nombres entiers et division</b>	<b>33</b>
5.1	Divisibilité . . . . .	33
5.2	Arithmétique modulaire . . . . .	34
5.3	Les fonctions pgcd et ppcm . . . . .	36
5.4	Algorithme d'Euclide . . . . .	37
5.5	Cryptographie . . . . .	38
<b>6</b>	<b>Démonstrations</b>	<b>41</b>
6.1	Règles d'inférence . . . . .	41
6.2	Démonstration directe . . . . .	42
6.3	Contraposée . . . . .	43
6.4	Démonstration par contradiction . . . . .	43
6.5	Cas par cas . . . . .	44

6.6	Équivalences . . . . .	45
<b>7</b>	<b>Algorithmes</b>	<b>47</b>
7.1	Généralités . . . . .	47
7.2	Pseudocode . . . . .	48
7.3	Le problème de la fouille . . . . .	48
7.4	Plus grand commun diviseur . . . . .	50
7.5	Palindromes . . . . .	51
<b>8</b>	<b>Complexité</b>	<b>53</b>
8.1	Calcul du maximum . . . . .	53
8.2	Palindromes . . . . .	54
8.3	Fouille binaire . . . . .	54
8.4	Différentes complexités . . . . .	55
8.5	Notation grand-O . . . . .	55
<b>9</b>	<b>Induction mathématique</b>	<b>59</b>
9.1	Principe de l'induction . . . . .	59
9.2	Démonstrations arithmétiques . . . . .	60
9.3	Démonstrations combinatoires . . . . .	61
9.4	Induction forte . . . . .	62
9.5	Application de l'induction forte . . . . .	62
<b>10</b>	<b>Récurtivité</b>	<b>65</b>
10.1	Exemples . . . . .	65
10.2	Récurtivité et induction . . . . .	67
10.3	Algorithmes récursifs . . . . .	68
<b>11</b>	<b>Relations</b>	<b>71</b>
11.1	Représentation . . . . .	71
11.2	Propriétés . . . . .	72
11.3	Relations d'équivalence . . . . .	74
11.4	Combinaisons de relations . . . . .	76
11.5	Clôture/fermeture des relations . . . . .	76

# Chapitre 1

## Logique

Commençons avec l'énigme suivante de R. Smullyan. Un chevalier se présente devant deux portes sur lesquelles sont inscrites les phrases suivantes :

- **Porte 1.** Derrière cette porte se trouve une princesse et derrière l'autre porte se trouve un dragon.
- **Porte 2.** Une de ces portes cache un dragon et une de ces portes cache une princesse.

Sachant qu'une des portes dit vrai et l'autre dit faux, et en supposant que le chevalier préfère la princesse au dragon, quelle porte devrait-il choisir ? Il est possible de répondre à cette question par un simple raisonnement, mais également de la modéliser à l'aide de la logique mathématique.

De façon plus générale, la logique est à la base de l'informatique :

- Elle est nécessaire pour écrire des conditions booléennes;
- Elle facilite les raisonnements;
- Sa simplicité et son expressivité permettent de modéliser de nombreux problèmes complexes.

### 1.1 Propositions

**Définition 1.** Une *proposition* est une expression qui est vraie ou fausse, mais pas les deux. On dit alors que sa *valeur de vérité* est *vrai* (1) ou *faux* (0) selon le cas.

**Exemple 1.** — “J’aime les mathématiques” est une proposition;

- “ $4! = 24$ ” est une proposition dont la valeur de vérité est *vrai*;
- “ $x + y = 2$ ” n’est pas une proposition, car sa valeur de vérité est indéterminée (elle dépend de la valeur de  $x$  et de  $y$ ).

On désigne souvent les propositions par les lettres  $p$ ,  $q$ ,  $r$  et  $s$ . Il est aussi intéressant de combiner des propositions.

**Définition 2.** Soit  $p$  une proposition. Alors

- la *négation* de  $p$ , notée  $\neg p$ , est la proposition qui vaut *vrai* si  $p$  est fausse, et qui vaut *faux* si  $p$  est vraie.
- la *conjonction* de  $p$  et  $q$ , notée  $p \wedge q$ , est la proposition qui est vraie si  $p$  et  $q$  sont vraies, fausse sinon.

- la *disjonction* de  $p$  et  $q$ , notée  $p \vee q$ , est la proposition qui est vraie si  $p$  ou  $q$  (ou les deux) sont vraies, fausse sinon.
- la *disjonction exclusive* de  $p$  et  $q$ , notée  $p \oplus q$ , est la proposition qui est vraie si  $p$  ou  $q$  (mais pas les deux) sont vraies, fausse sinon.
- l'*implication* de  $p$  vers  $q$ , notée  $p \rightarrow q$ , est la proposition qui est fausse si  $p$  est vraie et  $q$  est fausse, vraie sinon.
- la *biconditionnelle* entre  $p$  et  $q$ , notée  $p \leftrightarrow q$  est la proposition qui est vraie si  $p$  et  $q$  ont la même valeur de vérité, fausse sinon.

On appelle ces opérations sur les propositions des *connecteurs logiques*. Une façon visuelle de définir un connecteur logique est d'utiliser une *table de vérité*. Pour la négation, qui est un connecteur *unaire*, on a donc :

$p$	$\neg p$
0	1
1	0

Et pour les *connecteurs binaires*, on a :

$p$	$q$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \rightarrow q$	$p \leftrightarrow q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Dans tous les langages de programmation, les connecteurs  $\neg$ ,  $\wedge$ ,  $\vee$  et  $\oplus$  sont disponibles. Par exemple, en Java, il faut utiliser les symboles `!`, `&&`, `||` et `^` respectivement.

**Exemple 2.** Exprimons la proposition suivante à l'aide de propositions élémentaires et de connecteurs logiques :

”S’il pleut alors le trottoir n’est pas sec.”

On peut définir les propositions suivantes :

$p$  : ”il pleut”  
 $q$  : ”le trottoir est sec”

Alors la proposition correspond à  $p \rightarrow \neg q$ .

Nous pouvons aussi résoudre formellement l'énigme de Smullyan :

**Exemple 3.** Reprenons l'énigme de Smullyan.

- **Porte 1.** Derrière cette porte se trouve une princesse et derrière l'autre porte se trouve un dragon.
- **Porte 2.** Une de ces portes cache un dragon et une de ces portes cache une princesse.

On définit d'abord les propositions suivantes :

$$\begin{aligned} d_i & : \text{ "Il y a un dragon derrière la porte } i\text{", pour } i = 1, 2 \\ p_i & : \text{ "La porte } i \text{ dit vrai", pour } i = 1, 2 \end{aligned}$$

Alors on peut traduire les deux énoncés par

$$\begin{aligned} p_1 & \leftrightarrow \neg d_1 \wedge d_2 \\ p_2 & \leftrightarrow d_1 \oplus d_2 \end{aligned}$$

De plus, on sait qu'exactlyement une des deux portes dit vraie et l'autre fausse, ce qui se traduit par  $p_1 \oplus p_2$ . Faisons la table de vérité de  $(\neg d_1 \wedge d_2) \oplus (d_1 \oplus d_2)$ . On obtient

$d_1$	$d_2$	$\neg d_1$	$\neg d_1 \wedge d_2$	$d_1 \oplus d_2$	$(\neg d_1 \wedge d_2) \oplus (d_1 \oplus d_2)$
0	0	1	0	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	1	0	0	0	0

Le seul 1 de la dernière colonne se trouve dans la troisième ligne. Autrement dit, la seule façon d'avoir qu'exactlyement une des portes dise vrai et l'autre fausse, c'est que  $d_1$  soit vraie et  $d_2$  soit fausse, c'est-à-dire que la princesse se trouve derrière la porte 2.

Certaines propositions présentent un intérêt particulier.

**Définition 3.** Une *tautologie* est une proposition qui est toujours vraie. Une *contradiction* est une proposition qui est toujours fausse.

**Exemple 4.** La proposition  $p \vee \neg p$  est une tautologie. Il suffit de jeter un coup d'oeil à la table de vérité suivante pour le constater :

$p$	$\neg p$	$p \vee \neg p$
0	1	1
1	0	1

## 1.2 Équivalences logiques

Il est souvent fastidieux de remplir des tables de vérité pour calculer la valeur de vérité d'une expression complexe. Dans certaines situations, on préfère utiliser des équivalences logiques.

**Définition 4.** On dit que deux propositions  $p$  et  $q$  sont *logiquement équivalentes* si  $p \leftrightarrow q$  est une tautologie. On écrit alors  $p \Leftrightarrow q$ .

Une équivalence logique particulièrement utile permet de réécrire l'implication par rapport aux connecteurs  $\neg$  et  $\vee$ . En effet, on montre facilement que  $p \rightarrow q$  est logiquement équivalent à  $\neg p \vee q$  :

$p$	$q$	$\neg p$	$p \rightarrow q$	$\neg p \vee q$	$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	0	1	1	1

On a bien une tautologie, ce qui nous permet de conclure que les propositions sont logiquement équivalentes.

Il existe de nombreuses équivalences logiques pratiques :

Identité	$p \wedge V \Leftrightarrow p$ $p \vee F \Leftrightarrow p$
Absorption	$p \vee V \Leftrightarrow V$ $p \wedge F \Leftrightarrow F$
Idempotence	$p \vee p \Leftrightarrow p$ $p \wedge p \Leftrightarrow p$
Double négation	$\neg\neg p \Leftrightarrow p$
Commutativité	$p \vee q \Leftrightarrow q \vee p$ $p \wedge q \Leftrightarrow q \wedge p$
Associativité	$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$ $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$
Distributivité	$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$
Lois de Morgan	$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$ $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$

Les équivalences logiques permettent de simplifier des expressions complexes en expressions parfois beaucoup plus simples.

**Exemple 5.** Simplifiez le plus possible la proposition  $(p \wedge q) \rightarrow (p \vee q)$ .

On obtient

$$\begin{aligned}
 (p \wedge q) \rightarrow (p \vee q) &\Leftrightarrow \neg(p \wedge q) \vee (p \vee q) \\
 &\Leftrightarrow (\neg p \vee \neg q) \vee (p \vee q) \\
 &\Leftrightarrow (\neg p \vee p) \vee (q \vee \neg q) \\
 &\Leftrightarrow V \vee V \\
 &\Leftrightarrow V.
 \end{aligned}$$

Ainsi, il s'agit d'une tautologie.

## 1.3 Fonctions propositionnelles

Les propositions sont particulièrement intéressantes lorsqu'elles dépendent d'une variable. Par exemple, nous savons que  $x + y = 3$  n'est pas une proposition. En revanche, si  $x = 2$  et  $y = 2$ , on voit que l'énoncé devient faux.

**Définition 5.** On dit que  $P(x)$  est une *fonction propositionnelle* d'une variable (ou un *prédicat*) s'il s'agit d'une proposition lorsque la valeur de  $x$  est fixée.

On généralise naturellement cette définition aux fonctions propositionnelles de plusieurs variables.

**Exemple 6.** L'expression

- $P(x, y) : “x + y = 3”$  est une fonction propositionnelle. En particulier,  $P(1, 2)$  est vraie mais  $P(-3, 4)$  est fausse.
- $C(a, b, r, x, y) : “(x - a)^2 + (y - b)^2 \leq r^2”$  est une fonction propositionnelle qui prend la valeur vraie si et seulement si le point  $(x, y)$  se trouve à l’intérieur du disque de centre  $(a, b)$  de rayon  $r$ .

L’ensemble des valeurs que peuvent prendre les variables  $x, y$ , etc. est appelé *univers du discours* (ou *domaine*) de la fonction propositionnelle.

## 1.4 Quantificateurs

Dans la logique du premier ordre, il existe deux quantificateurs permettant de former de nouvelles propositions.

**Définition 6.** Soit  $P(x)$  une fonction propositionnelle.

- La *quantification universelle* de  $P(x)$ , notée  $\forall x P(x)$ , est la proposition qui prend la valeur vraie si et seulement si  $P(x)$  est vraie pour tout  $x$  dans l’univers du discours;
- La *quantification existentielle* de  $P(x)$ , notée  $\exists x P(x)$ , est la proposition qui prend la valeur vraie si et seulement si  $P(x)$  est vraie pour au moins une valeur de  $x$  dans l’univers du discours.

**Exemple 7.** Voici quelques exemples :

- Soit  $P(x) : “x + 1 > x”$  et supposons que l’univers du discours est l’ensemble des nombres réels. Alors  $\forall x P(x)$  est vraie.
- Soit  $P(x) : “5x = 4”$ . Si l’univers du discours est l’ensemble des nombres naturels, alors  $\exists x P(x)$  est fausse, mais si l’univers du discours est l’ensemble des nombres rationnels (ou réels), alors  $\exists x P(x)$  est vraie.

On peut représenter plusieurs énoncés grâce à ces quantificateurs

**Exemple 8.** Considérons les prédicats suivants, où l’univers du discours est l’ensemble des animaux :

$$\begin{aligned} M(x, y) &: “x \text{ chasse } y” \\ H(x) &: “x \text{ est herbivore}” \\ C(x) &: “x \text{ est carnivore}” \\ O(x) &: “x \text{ est omnivore}” \end{aligned}$$

Traduisons les énoncés suivants :

- Il existe un animal qui chasse le poisson :

$$\exists x M(x, \text{poisson})$$

- Tout animal est soit herbivore, soit carnivore ou soit omnivore :

$$\forall x (H(x) \vee C(x) \vee O(x)) \wedge \neg (H(x) \wedge C(x)) \wedge \neg (H(x) \wedge O(x)) \wedge \neg (C(x) \wedge O(x))$$

- Tout animal carnivore chasse un autre animal :

$$\forall x C(x) \rightarrow \exists y M(x, y)$$

À noter qu'on peut également écrire cette proposition sous la forme  $\forall x \exists y C(x) \rightarrow M(x, y)$ .

- Il existe un carnivore qui chasse tous les autres animaux :

$$\exists x \forall y C(x) \wedge M(x, y)$$

## 1.5 Applications

En géométrie computationnelle, il existe plusieurs prédicats permettant de tester rapidement si des objets se touchent, sont en collision, etc.

- Est-ce que deux objets circulaires sont en collision ?

$$\text{Collision}(O_1, O_2) : (O_1.x - O_2.x)^2 + (O_1.y - O_2.y)^2 \leq O_1.r^2 + O_2.r^2$$

- Est-ce que deux intervalles se chevauchent ?

$$\begin{aligned} \text{SeChevauchent}(I_1, I_2) : & (I_1.\text{gauche} \leq I_2.\text{gauche} \leq I_1.\text{droit} \leq I_2.\text{droit}) \vee \\ & (I_2.\text{gauche} \leq I_1.\text{gauche} \leq I_2.\text{droit} \leq I_1.\text{droit}) \end{aligned}$$

Une autre application concerne le langage de programmation Prolog, basé sur la logique du premier ordre, développé dans les années 70. Il permet par exemple de modéliser la célèbre énigme des cinq maisons, dont l'idée est attribuée à Einstein :

```
/*
L'énigme d'Einstein (ou l'énigme des cinq maisons), attribuée au physicien
Einstein s'énonce comme suit :
```

```
Il y a cinq maisons de cinq couleurs différentes. Dans chacune de ces maisons
vit une personne de nationalité différente. Chacune de ces personnes boit une
boisson différente, fume un cigare différent et a un animal domestique
différent.
```

1. L'Anglais vit dans la maison rouge.
2. Le Suedois a des chiens.
3. Le Danois boit du thé.
4. La maison verte est à gauche de la maison blanche.
5. Le propriétaire de la maison verte boit du café.
6. La personne qui fume des Pall Mall a des oiseaux.
7. Le propriétaire de la maison jaune fume des Dunhill.
8. La personne qui vit dans la maison du centre boit du lait.
9. Le Norvégien habite dans la première maison.
10. L'homme qui fume des Blend vit à côté de celui qui a des chats.
11. L'homme qui a un cheval est le voisin de celui qui fume des Dunhill.
12. Le propriétaire qui fume des Blue Master boit de la bière.
13. L'Allemand fume des Prince.
14. Le Norvégien vit juste à côté de la maison bleue.
15. L'homme qui fume des Blend a un voisin qui boit de l'eau.

Question : qui a le poisson ?

On represente une maison par une liste de la forme  
 [couleur,nationalite,boisson,cigare,animal]

\*/

```
gauche(X,Y,L) :- append(_,[X,Y|_],L).
```

```
voisin(X,Y,L) :- gauche(X,Y,L) ; gauche(Y,X,L).
```

```
resoudre(Maisons) :- length(Maisons,5),
    member([rouge,anglais,_,_,_],Maisons),
    member([_,suedois,_,_,chien],Maisons),
    member([_,danois,the,_,_],Maisons),
    gauche([verte,_,_,_,_],[blanche,_,_,_,_],Maisons),
    member([verte,_,cafe,_,_],Maisons),
    member([_,_,_,pallmall,oiseaux],Maisons),
    member([jaune,_,_,dunhill,_],Maisons),
    Maisons=[_,_,[_],_,lait,_,_,_],
    Maisons=[[_],norvegien,_,_,_,_,_,_,_],
    voisin([_,_,_,blend,_],[_,_,_,_,_,chat],Maisons),
    voisin([_,_,_,_,cheval],[_,_,_,dunhill,_],Maisons),
    member([_,_,biere,bluemaster,_],Maisons),
    member([_,allemand,_,prince,_],Maisons),
    voisin([_,norvegien,_,_,_],[bleue,_,_,_,_],Maisons),
    voisin([_,_,_,blend,_],[_,_,eau,_,_,_],Maisons),
    member([_,_,_,_,poisson],Maisons).
```



# Chapitre 2

## Ensembles

Nous commençons par une définition.

**Définition 7.** Un *ensemble* est une collection d'objets appelés *éléments*.

Souvent, on dénote les ensembles par des lettres majuscules  $A$ ,  $B$ ,  $C$ , etc. et les éléments par des lettres minuscules  $a$ ,  $b$ ,  $c$ , etc.

Il y a essentiellement deux façons de représenter des ensembles :

— Par *extension*, par exemple

$$\begin{aligned} P &= \{\text{Alice, Bob, Carl}\} \\ \mathbb{N} &= \{0, 1, 2, \dots\} \text{ l'ensemble des nombres naturels} \\ \mathbb{N}^* &= \{1, 2, \dots\} \\ \mathbb{Z} &= \{\dots, -2, -1, 0, 1, 2, \dots\} \text{ l'ensemble des nombres entiers} \\ \mathbb{Z}^* &= \{\dots, -2, -1, 1, 2, \dots\} \\ \mathbb{Q} &= \left\{ 0, 1, 2, \dots, -1, -2, -3, \dots, \frac{1}{2}, \frac{1}{3}, \dots, -\frac{1}{2}, -\frac{2}{3}, \dots \right\} \end{aligned}$$

— Par *compréhension*, par exemple

$$\begin{aligned} A &= \{0, 2, 4, 6, 8, \dots, 100\} \\ &= \{a \in \mathbb{N} \mid 0 \leq a \leq 100 \wedge a \text{ est pair}\} \\ \mathbb{Q} &= \left\{ \frac{a}{b} \mid a \in \mathbb{Z} \wedge b \in \mathbb{Z}^* \right\} \end{aligned}$$

Si l'élément appartient à l'ensemble  $A$ , on écrit  $a \in A$ , sinon, on écrit  $a \notin A$ .

### 2.1 Relations ensemblistes

**Définition 8.** Soient  $A$  et  $B$  deux ensembles. Alors on dit que

- $A$  est un *sous-ensemble* de  $B$ , et on écrit  $A \subseteq B$ , si pour tout  $a \in A$ , on a  $a \in B$ . Autrement dit, la proposition  $a \in A \rightarrow a \in B$  doit être satisfaite;
- $A$  est *égal* à  $B$ , noté  $A = B$ , si  $A \subseteq B$  et  $B \subseteq A$ .

**Exemple 9.** Considérons les ensembles

$$A = \{1, 3, 5, 7, 9\}, \quad B = \{5, 3, 1, 7, 7, 9\}, \quad C = \{3, 5, 9\} \quad \text{et} \quad D = \{7, 7, 1, 3\}.$$

Alors on a les relations suivantes

$$A = B, \quad C \subseteq A, \quad D \subseteq B.$$

De plus, si

$$E = \{a \in \mathbb{Z} \mid 0 \leq a \leq 10 \wedge a \text{ est impair}\},$$

alors on a  $A = E$ .

Nous avons aussi les inclusions  $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$ . Un ensemble particulièrement important est celui qui ne contient aucun élément.

**Définition 9.** L'*ensemble vide*, noté  $\emptyset$ , est l'ensemble n'ayant aucun élément.

Peu importe l'ensemble  $A$ , nous avons toujours les relations  $\emptyset \subseteq A$  et  $A \subseteq A$ .

Dans certains cas, on s'intéresse à l'énumération de tous les sous-ensembles d'un ensemble donné.

**Définition 10.** Soit  $A$  un ensemble. Alors l'*ensemble des parties* de  $A$ , noté  $\mathcal{P}(A)$  ou  $2^A$  est l'ensemble de tous les sous-ensembles de  $A$ .

**Exemple 10.** Soit  $A = \{a, b, c\}$ . Alors

$$2^A = \mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}.$$

Aussi,

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

## 2.2 Opérations ensemblistes

Souvent, on s'intéresse au nombre d'éléments présents dans un ensemble.

**Définition 11.** Soit  $A$  un ensemble. Alors la *cardinalité* de  $A$  est le nombre d'éléments de  $A$ . On la note  $|A|$  ou  $\#(A)$ . Elle peut être finie ou infinie.

Voici quelques exemples :

$$\begin{aligned} |\{a, b, g, \ell\}| &= 4 \\ |\emptyset| &= 0 \\ |\mathbb{Z}| &= +\infty \end{aligned}$$

L'ordre des éléments d'un ensemble n'est pas important. Il existe cependant une notion permettant de marquer l'ordre dans lequel sont présentés les éléments.

**Définition 12.** Soit  $n \geq 1$  un entier et  $A$  un ensemble. On dit que  $(a_1, a_2, \dots, a_n)$  est un *n-tuplet* de  $A$  si  $a_i \in A$  pour  $i = 1, 2, \dots, n$ .

Un  $n$ -tuplet est en quelque sorte l'homologue mathématiques des tableaux informatiques. Deux  $n$ -tuplets sont donc égaux seulement si les éléments correspondants le sont, en tenant compte de l'ordre.

**Exemple 11.** On a  $\{1, 2, 3\} = \{1, 3, 2\}$ , mais  $(1, 2, 3) \neq (1, 3, 2)$ .

Cela nous mène naturellement à la définition suivante :

**Définition 13.** Soient  $A$  et  $B$  deux ensembles. Alors le *produit cartésien* de  $A$  et  $B$ , noté  $A \times B$ , est l'ensemble de tous les couples  $(a, b)$ , où  $a \in A$  et  $b \in B$ , c'est-à-dire

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}.$$

**Exemple 12.** On a

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

Le produit cartésien s'étend à plus de deux ensembles.

**Définition 14.** Soient  $n \geq 2$  un entier et  $A_1, A_2, \dots, A_n$ . Alors

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) \mid a_i \in A_i, \text{ pour } i = 1, 2, \dots, n\}.$$

**Exemple 13.** On a

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

L'ensemble des points du plan cartésien est dénoté par  $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$ . L'ensemble des points de l'espace euclidien est  $\mathbb{R}^3$ .

## 2.3 Autres opérations

À l'instar des propositions logiques, il est possible de combiner les ensembles à l'aide d'opération sur ceux-ci :

**Définition 15.** Soient  $A$  et  $B$  deux sous-ensembles d'un univers  $U$  (ajouter diagrammes de Venn).

- La *réunion* de  $A$  et  $B$ , notée  $A \cup B$ , est l'ensemble qui contient un élément  $a$  si et seulement si  $a \in A$  ou  $a \in B$  (ou les deux), c'est-à-dire

$$A \cup B = \{x \in U \mid x \in A \vee x \in B\}.$$

- L'*intersection* de  $A$  et  $B$ , notée  $A \cap B$ , est l'ensemble qui contient un élément  $a$  si et seulement si  $a \in A$  et  $a \in B$ , c'est-à-dire

$$A \cap B = \{x \in U \mid x \in A \wedge x \in B\}.$$

- Le *complément* de  $A$ , noté  $\overline{A}$ , est l'ensemble qui contient un élément  $a$  si et seulement si  $a \notin A$ , c'est-à-dire

$$\overline{A} = \{x \in U \mid x \notin A\}.$$

- La *différence* entre  $A$  et  $B$ , notée  $A - B$ , est l'ensemble qui contient un élément  $a$  si et seulement si  $a \in A$ , mais  $a \notin B$ , c'est-à-dire

$$A - B = \{x \in U \mid x \in A \wedge x \notin B\}.$$

- La *différence symétrique* entre  $A$  et  $B$ , notée  $A \oplus B$ , est l'ensemble qui contient un élément  $a$  si et seulement si  $a \in A$  ou  $a \in B$ , mais pas les deux, c'est-à-dire

$$A \oplus B = \{x \in U \mid x \in A \oplus x \in B\}.$$

En outre, si  $A \cap B = \emptyset$ , on dit que  $A$  et  $B$  sont *disjoints*.

**Exemple 14.** Soient  $U$  l'ensemble des chiffres entre 0 et 9,

$$A = \{1, 2, 4, 5, 6, 8\} \quad \text{et} \quad B = \{2, 3, 6, 7, 9\}.$$

Alors

$$\begin{aligned} A \cup B &= \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \\ A \cap B &= \{2, 6\} \\ \overline{A} &= \{0, 3, 7, 9\} \\ A - B &= \{1, 4, 5, 8\} \\ A \oplus B &= \{1, 3, 4, 5, 7, 8, 9\} \end{aligned}$$

**Exemple 15.** Soient  $a$  et  $b$  deux entiers. On définit l'ensemble

$$a\mathbb{Z} + b = \{ax + b \mid x \in \mathbb{Z}\}.$$

Par exemple,  $3\mathbb{Z} = 3\mathbb{Z} + 0 = \{3x \mid x \in \mathbb{Z}\} = \{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\}$  dénote l'ensemble des multiples de 3. Alors

$$\begin{aligned} (2\mathbb{Z} + 4) \cap (4\mathbb{Z} + 2) &= \{\dots, -4, -2, 0, 2, 4, \dots\} \cap \{\dots, -6, -2, 2, 6, \dots\} \\ &= \{\dots, -6, -2, 2, 6, \dots\} \\ &= 4\mathbb{Z} + 2 \\ 3\mathbb{Z} - 6\mathbb{Z} &= \{\dots, -6, -3, 0, 3, 6, \dots\} - \{\dots, -12, -6, 0, 6, 12, \dots\} \\ &= \{\dots, -9, -3, 3, 9, \dots\} \\ &= 6\mathbb{Z} + 3 \\ 3\mathbb{Z} \cap 5\mathbb{Z} &= \{\dots, -9, -6, -3, 0, 3, 6, \dots\} \cap \{\dots, -15, -10, -5, 0, 5, 10, \dots\} \\ &= \{\dots, -30, -15, 0, 15, 30, \dots\} \\ &= 15\mathbb{Z} \\ 4\mathbb{Z} \oplus 8\mathbb{Z} &= \{\dots, -12, -4, 4, 12, \dots\} \\ &= 8\mathbb{Z} + 4. \end{aligned}$$

Les équivalences logiques vues au chapitre précédent se traduisent directement en propriétés sur les ensembles :

Identité	$A \cup \emptyset = A$ $A \cap U = A$
Absorption	$A \cup U = U$ $A \cap \emptyset = \emptyset$
Idempotence	$A \cup A = A$ $A \cap A = A$
Double négation	$\overline{\overline{A}} = A$
Commutativité	$A \cup B = B \cup A$ $A \cap B = B \cap A$
Associativité	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributivité	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Lois de Morgan	$\overline{A \cup B} = \overline{A} \cap \overline{B}$ $\overline{A \cap B} = \overline{A} \cup \overline{B}$

**Proposition 1.** Soient  $A$  et  $B$  deux ensembles. Alors

$$\begin{aligned}
 |A \times B| &= |A||B| \\
 |2^A| &= 2^{|A|} \\
 |A \cup B| &= |A| + |B| - |A \cap B| \\
 |\overline{A}| &= |U| - |A| \\
 |A - B| &= |A| - |A \cap B| \\
 |A \oplus B| &= |A| + |B| - 2|A \cap B|.
 \end{aligned}$$

En outre,

- si  $A \subseteq B$ , alors  $|A| \leq |B|$  et
- si  $A = B$ , alors  $|A| = |B|$ .

**Exemple 16.** Soient  $U$  un ensemble et  $A, B, C \subseteq U$  trois sous-ensembles de  $U$ . Supposons que les huit conditions suivantes sont vérifiées :

1.  $|\mathcal{P}(A \cap B \cap C)| = 256$ ;
2.  $|B \cap C| = 15$ ;
3.  $|A \cap C| = 14$ ;
4.  $|A \cap B| = 13$ ;
5.  $|B| = |C|$ ;
6.  $|B \oplus C| = 16$ ;
7.  $|A \times B| = 529$ ;
8.  $|\overline{A} \cap \overline{B} \cap \overline{C}| = |\mathcal{P}(\emptyset)|$ ;

Calculons la valeur de  $|U|$ .

## 2.4 Opérations généralisées

Dans certaines situations, on s'intéresse au calcul de l'intersection ou de la réunion de plusieurs ensembles :

**Définition 16.** Soient  $A_1, A_2, \dots, A_n$  des ensembles. Alors

$$\bigcup_{i=1}^n A_i = A_1 \cup A_2 \cup \dots \cup A_n$$

$$\bigcap_{i=1}^n A_i = A_1 \cap A_2 \cap \dots \cap A_n.$$

**Exemple 17.** Si on reprend la notation  $a\mathbb{Z} + b$ , alors

$$\bigcup_{i=0}^7 (8\mathbb{Z} + i) = \mathbb{Z}$$

$$\bigcap_{i=2}^5 i\mathbb{Z} = 60\mathbb{Z}.$$

Nous terminons en discutant de la notion de partition.

**Définition 17.** Soit  $A$  un ensemble et  $A_i \subseteq A$ , pour  $i = 1, 2, \dots, n$ . On dit que  $\{A_i \mid i = 1, 2, \dots, n\}$  est une *partition* de  $A$  si

1. (couverture)  $\bigcup_{i=1}^n A_i = A$ ;
2. (disjonction) si  $A_i \cap A_j \neq \emptyset$ , alors  $i = j$ .

**Exemple 18.** Considérons les exemples suivants.

- Soit  $A$  l'alphabet latin de 26 lettres,  $V$  l'ensemble des voyelles et  $C$  l'ensemble des consonnes. Alors  $\{V, C\}$  est une partition de  $A$ .
- L'ensemble  $\{3\mathbb{Z}, 3\mathbb{Z} + 1, 3\mathbb{Z} + 2\}$  est une partition de  $\mathbb{Z}$ .

# Chapitre 3

## Fonctions

Les fonctions permettent d'établir une correspondance entre des éléments de différents ensembles.

### 3.1 Définitions de base

**Définition 18.** Une *fonction* est un triplet  $f = (A, B, G)$  où

- $A$  est un ensemble, appelé *ensemble de départ* ou *domaine*, noté  $\text{dom}(f)$ ;
- $B$  est un ensemble, appelé *ensemble d'arrivée* ou *codomaine*, noté  $\text{codom}(f)$ ;
- $G \subseteq A \times B$  tel que pour tout  $a \in A$ , il existe un unique  $b \in B$  tel que  $(a, b) \in G$ . L'ensemble  $G$  est appelé *graphe* de  $f$ .

Plutôt que d'écrire  $f = (A, B, G)$ , on utilise en général la notation  $f : A \rightarrow B$  et on décrit  $G$  à l'aide d'une *règle de correspondance*. Finalement, si  $f(a) = b$ , alors on dit que  $b$  est l'*image* de  $a$  et que  $a$  est une *préimage* de  $b$ .

**Exemple 19.** Soit la fonction  $f : \{a, b, c\} \rightarrow \{1, 2\}$  définie par  $f(a) = 2$ ,  $f(b) = 1$  et  $f(c) = 1$ . Alors

- $\text{dom}(f) = \{a, b, c\}$ ;
- $\text{codom}(f) = \{1, 2\}$ ;
- Le graphe de  $f$  est  $\{(a, 2), (b, 1), (c, 1)\}$ ;
- L'image de  $c$  est 1. Les préimages de 1 sont  $b$  et  $c$ .

**Exemple 20.** Soit  $f : \mathbb{R} \rightarrow \mathbb{R}$  la fonction telle que  $f(x) = x^2$ . Alors

- Le domaine et le codomaine sont  $\mathbb{R}$ ;
- Le graphe de  $f$  est  $\{(x, x^2) \mid x \in \mathbb{R}\}$ ;
- L'image de  $-2$  est 4;
- Les préimages de 7 sont  $\pm\sqrt{7}$ .

**Définition 19.** L'*ensemble image* d'une fonction  $f$  est l'ensemble des images de  $f : A \rightarrow B$ , c'est-à-dire

$$f(A) = \{b \in B \mid \exists a \in A, b = f(a)\}.$$

## 3.2 Fonctions usuelles

**Exemple 21.** La *fonction plancher*  $\lfloor \cdot \rfloor : \mathbb{R} \rightarrow \mathbb{R}$  est la fonction qui associe à chaque nombre réel  $x$  l'entier le plus près de  $x$  qui est plus petit que  $x$ . Par exemple

$$\lfloor \pi \rfloor = 3, \quad \left\lfloor \frac{3}{2} \right\rfloor = 1, \quad \lfloor 5 \rfloor = 5 \quad \text{et} \quad \lfloor -1.3 \rfloor = -2.$$

Il est facile de voir que  $f(\mathbb{R}) = \mathbb{Z}$ .

Les opérations arithmétiques sont toutes des fonctions :

- $+$  :  $\mathbb{R}^2 \rightarrow \mathbb{R}$
- $-$  :  $\mathbb{R}^2 \rightarrow \mathbb{R}$
- $\cdot$  :  $\mathbb{R}^2 \rightarrow \mathbb{R}$
- $/$  :  $\mathbb{R} \times \mathbb{R}^* \rightarrow \mathbb{R}$
- $/$  :  $\mathbb{R} \times \mathbb{R}^* \rightarrow \mathbb{R}$

Par contre, on utilise la notation *infixe* dans leur cas plutôt que la notation préfixe. De la même façon, les connecteurs logiques et les opérations ensemblistes sont des *fonctions*.

**Définition 20.** Soit  $A$  un ensemble. La *fonction identité* sur  $A$  est la fonction  $\text{Id} : A \rightarrow A : a \mapsto a$ .

Des fonctions qu'on rencontre souvent en informatique sont celles définies *par morceaux*, c'est-à-dire que la règle de correspondance varie selon qu'une condition (booléenne) est satisfaite ou non.

**Exemple 22.** La fonction *valeur absolue* est définie par

$$| \cdot | : \mathbb{R} \rightarrow \mathbb{R} : x \mapsto \begin{cases} x, & \text{si } x \geq 0; \\ -x, & \text{si } x < 0. \end{cases}$$

Ainsi,  $|11| = 11$ , puisque  $11 \geq 0$ , mais  $|-4| = -(-4) = 4$  puisque  $-4 < 0$ .

D'autres fonctions sont souvent utilisées en informatique :

- Les fonctions *polynomiales*

$$f(x) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_2 x^2 + c_1 x + c_0,$$

où les nombres  $c_i$  ( $i = 0, 1, 2, \dots, n$ ) sont appelés *coefficients*.

- Les fonctions *rationnelles*

$$f(x) = \frac{p(x)}{q(x)}$$

où  $p(x)$  et  $q(x)$  sont des fonctions polynomiales.

- Les fonctions *exponentielles*

$$f(x) = a^x$$

où  $a \in \mathbb{R}_+^*$ .

- Les fonctions *logarithmiques*

$$f(x) = \log_b(x)$$

où  $b \in \mathbb{R}_+^*$ .

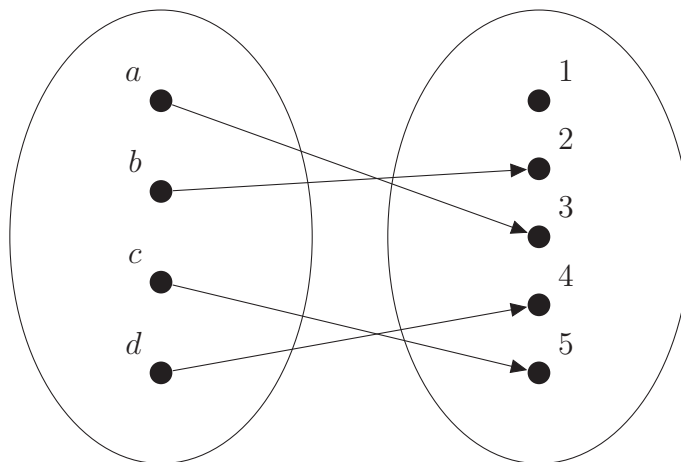
Il existe d'autres familles de fonctions importantes (les fonction trigonométriques par exemple), mais elles ne seront pas abordées dans ce cours. Bien entendu, ces fonctions peuvent être combinées ensemble pour former d'autres fonctions plus complexes.

### 3.3 Injectivité, surjectivité, bijectivité

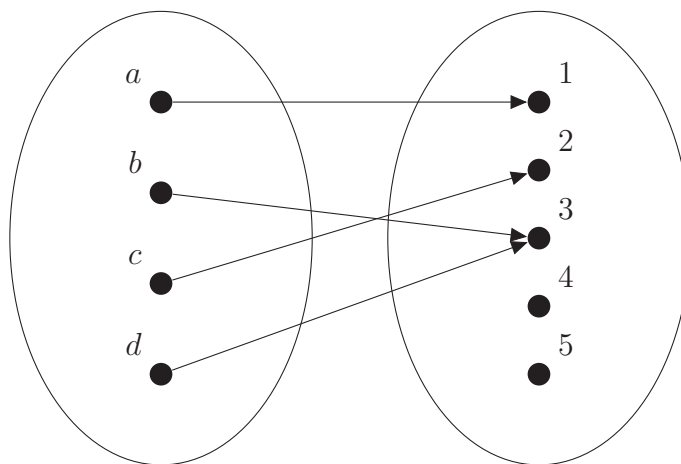
Certains types de fonctions présentent un intérêt particulier en mathématiques et en informatique. Parmi celles-ci, on compte les fonctions injectives :

**Définition 21.** Une fonction  $f : A \rightarrow B$  est dite *injective* (c'est donc une *injection*) s'il n'existe aucune paire d'éléments distincts ayant la même image. Plus formellement, si  $a_1, a_2 \in A$  et que  $a_1 \neq a_2$ , alors  $f(a_1) \neq f(a_2)$ . De façon équivalente, si  $f(a_1) = f(a_2)$ , alors nécessairement  $a_1 = a_2$ .

**Exemple 23.** La fonction suivante est injective :



Par contre, la fonction ci-bas ne l'est pas, car  $f(b) = f(d)$  :



**Exemple 24.** La fonction  $f : \mathbb{R}^+ \rightarrow \mathbb{R} : x \mapsto \sqrt{x}$  est injective. En effet, soient  $x$  et  $y$  deux nombres réels positifs tels que  $f(x) = f(y)$ . Alors  $\sqrt{x} = \sqrt{y}$  et donc  $x = y$ . Or,  $x, y \geq 0$ , ce qui entraîne  $x = y$ . On en conclut que  $f$  est injective.

**Exemple 25.** La fonction  $\lfloor \cdot \rfloor$  n'est pas injective. Par exemple,  $\lfloor 3/5 \rfloor = 0$  et  $\lfloor 1/4 \rfloor = 0$ , mais  $3/5 \neq 1/4$ .

**Exemple 26.** La rotation  $R : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  définie par

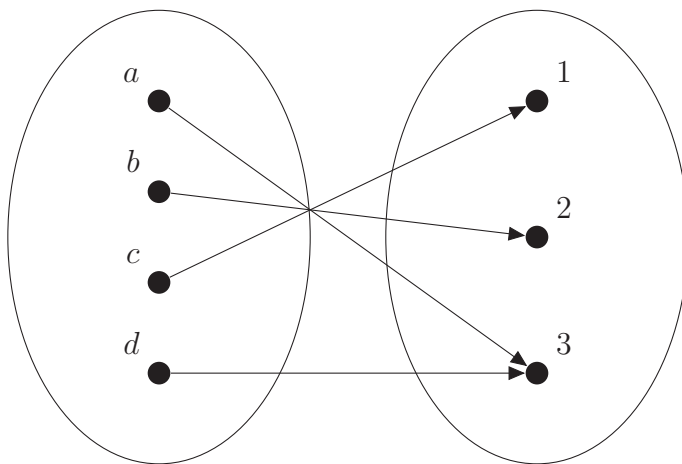
$$R(x, y) = (-y, x)$$

est injective. En effet, supposons qu'on ait des couples de nombres réels  $(x, y)$  et  $(z, t)$  tels que  $R(x, y) = R(z, t)$ . Alors  $(-y, x) = (-t, z)$  et donc  $-y = -t$  et  $x = z$ , ce qui entraîne  $(x, y) = (z, t)$ .

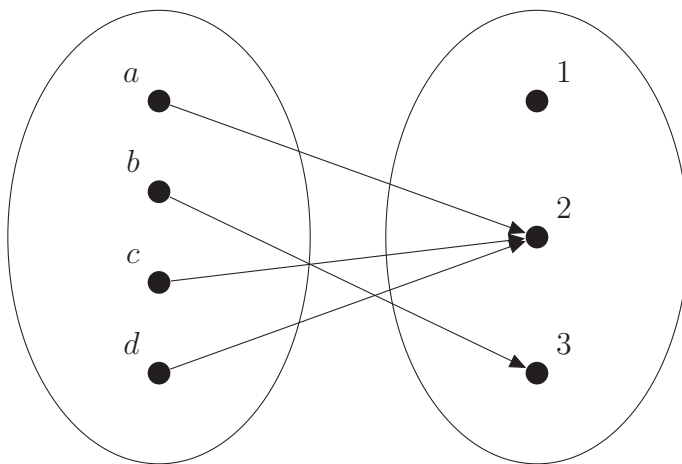
**Exemple 27.** Soit  $\mathcal{E}$  l'ensemble de tous les ensembles finis et  $\text{Card} : \mathcal{E} \rightarrow \mathbb{N}$  la fonction définie par  $\text{Card}(E) = |E|$ . La fonction  $\text{Card}$  n'est pas injective, car, par exemple,  $\text{Card}(\{0\}) = 1 = \text{Card}(\{\pi\})$  et, clairement,  $1 \neq \pi$ .

**Définition 22.** Une fonction  $f : A \rightarrow B$  est dite *surjective* si tout élément du codomaine  $B$  a au moins une préimage. Plus formellement, pour tout  $b \in B$ , il existe  $a \in A$  tel que  $b = f(a)$ .

**Exemple 28.** La fonction suivante est surjective :



Mais pas celle-ci, car il n'existe aucun  $a \in A$  tel que  $f(a) = 1$  :



**Exemple 29.** La fonction  $f : \mathbb{N} \rightarrow \mathbb{N} : x \mapsto x^2$  n'est pas surjective, car il n'existe aucun  $x \in \mathbb{N}$  tel que  $x^2 = 3$ .

En revanche, la fonction  $f : \mathbb{R} \rightarrow \mathbb{N} : x \mapsto x^2$  est surjective, puisque pour tout  $n \in \mathbb{N}$ , on a que  $\sqrt{n} \in \mathbb{R}$  et donc  $f(\sqrt{n}) = n$ .

**Exemple 30.** La fonction plancher  $\lfloor \cdot \rfloor : \mathbb{R} \rightarrow \mathbb{Z}$  est surjective. En effet, pour tout  $z \in \mathbb{Z}$ , on a  $\lfloor z \rfloor = z$ , c'est-à-dire que  $z$  est atteinte par lui-même.

**Exemple 31.** La fonction  $f : \mathbb{Z} \rightarrow \mathbb{Z} : x \mapsto 2x + 1$  n'est pas surjective. En effet, on remarque que  $f(\mathbb{Z}) = 2\mathbb{Z} + 1$  (l'ensemble des nombres impairs). Par conséquent, les nombres pairs ne peuvent être atteints. Par exemple, il n'existe aucun  $x \in \mathbb{Z}$  tel que  $f(x) = 0$ . Si c'était le cas, alors on aurait  $0 = 2x + 1$ , et donc  $x = -1/2 \notin \mathbb{Z}$ .

**Exemple 32.** La rotation  $R : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  définie par

$$R(x, y) = (-y, x)$$

est surjective. En effet, prenons  $(x, y) \in \mathbb{R}^2$ . Nous allons montrer qu'il existe  $(z, t) \in \mathbb{R}^2$  tels que  $R(z, t) = (x, y)$ . Il faut donc avoir  $(-t, z) = (x, y)$  et alors  $z = y$  et  $t = -x$ . Ainsi, il suffit de prendre  $(z, t) = (y, -x)$ . On a montré que peu importe  $(x, y) \in \mathbb{R}^2$ , on a  $R(y, -x) = (x, y)$ , ce qui montre que  $R$  est surjective.

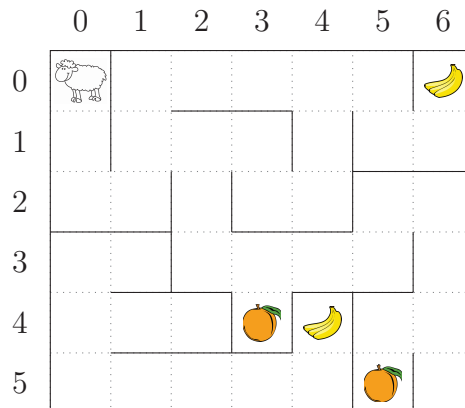
**Exemple 33.** La fonction  $\text{Card} : \mathcal{E} \rightarrow \mathbb{N}$  est surjective. En effet, pour tout  $n \in \mathbb{N}$ , on peut facilement trouver un ensemble ayant la cardinalité  $n$ . Par exemple, on a que  $\text{Card}(\emptyset) = 0$  et  $\text{Card}(\{1, 2, \dots, n\}) = n$ , pour tout  $n \geq 1$ .

**Définition 23.** Une fonction est dite *bijective* si elle est injective et surjective.

**Exemple 34.** Revenons sur les exemples précédents.

- La fonction plancher n'est pas bijective, car elle n'est pas injective;
- La fonction  $f(x) = x^2$  n'est bijective que si  $\text{dom}(f) = \mathbb{R}^+$  et  $\text{codom}(f) = \mathbb{R}^+$ .
- La rotation  $R : \mathbb{R}^2 \rightarrow \mathbb{R}^2 : (x, y) \mapsto (-y, x)$  est bijective. En fait, on peut montrer que toute translation, réflexion, rotation et symétrie glissée sont toutes des bijections dans  $\mathbb{R}^2$  et même dans  $\mathbb{R}^3$ .

**Exemple 35.** Soit le labyrinthe décrit par l'image ci-bas :



On représente une case du labyrinthe par un couple  $(i, j)$ , où  $i \in \{0, 1, 2, 3, 4, 5\}$  et  $j \in \{0, 1, 2, 3, 4, 5, 6\}$ . Aussi, on a les huit ensembles qui suivent :

$$\begin{aligned}
 \mathbb{N} &= \{0, 1, 2, 3, 4, \dots\} \\
 A &= \left\{ \text{banana}, \text{orange} \right\} \\
 B &= \{\text{vrai}, \text{faux}\} \\
 B^4 &= B \times B \times B \times B \\
 L &= \{0, 1, 2, 3, 4, 5\} \\
 C &= \{0, 1, 2, 3, 4, 5, 6\} \\
 L \times C &= \{(i, j) \mid i \in L, j \in C\} \\
 \mathcal{P}(L \times C) &= \{S \mid S \subseteq L \times C\}
 \end{aligned}$$

Donnons le domaine et le codomaine des fonctions suivantes. Indiquons également s'il s'agit d'injection, surjection et/ou bijection.

- La fonction qui indique, pour chaque case  $(i, j)$ , s'il existe un chemin à partir du mouton jusqu'à la case  $(i, j)$ ;
- La fonction qui, à chaque case  $(i, j)$  du labyrinthe, lui associe un quadruplet  $(p, q, r, s)$  indiquant si on peut se déplacer d'une case vers la droite, vers le haut, vers la gauche et vers le bas à partir de la case  $(i, j)$ .
- La fonction qui indique, pour chaque type de fruit, l'ensemble des cases où on le retrouve;
- La fonction qui indique, pour chaque colonne, le nombre de bananes qu'on y trouve.

### 3.4 Composition de fonctions

**Définition 24.** Soient  $f : A \rightarrow B$  et  $g : B \rightarrow C$  deux fonctions. La *composition* de  $g$  et  $f$ , notée  $g \circ f$  est la fonction

$$g \circ f : A \rightarrow C : a \mapsto g(f(a)).$$

**Exemple 36.** Soient  $f, g : \mathbb{R} \rightarrow \mathbb{R}$  définies par  $f(x) = x^2 + 1$  et  $g(x) = 2x + 1$ . Alors  $f \circ g : \mathbb{R} \rightarrow \mathbb{R}$  est définie par

$$(f \circ g)(x) = f(g(x)) = f(2x + 1) = (2x + 1)^2 + 1 = 4x^2 + 4x + 1 + 1 = 4x^2 + 4x + 2.$$

D'autre part,  $g \circ f : \mathbb{R} \rightarrow \mathbb{R}$  est définie par

$$(g \circ f)(x) = g(f(x)) = g(x^2 + 1) = 2(x^2 + 1) + 1 = 2x^2 + 3.$$

**Proposition 2.** Soit  $f : A \rightarrow A$  une fonction sur un ensemble  $A$ . Alors  $f \circ \text{Id} = \text{Id} \circ f = f$ .

**Exemple 37.** Soit  $R$  la rotation de  $90^\circ$  en sens antihoraire. Alors  $R(x, y) = (-y, x)$ . De plus, on a

$$\begin{aligned} R^2(x, y) &= (R \circ R)(x, y) \\ &= R(R(x, y)) \\ &= R(-y, x) \\ &= (-x, -y) \\ R^3(x, y) &= (R \circ R \circ R)(x, y) \\ &= R(-x, -y) \\ &= (y, -x) \\ R^4(x, y) &= R(y, -x) \\ &= (x, y) \end{aligned}$$

Ainsi,  $R^4 = \text{Id}$  sur  $\mathbb{R}^2$ .

**Définition 25.** Soit  $f : A \rightarrow B$  une bijection. La *fonction inverse* (ou *réciproque*) de  $f$ , notée  $f^{-1}$  est la fonction

$$f^{-1} : B \rightarrow A$$

telle que  $f^{-1} \circ f = \text{Id}_A$  et  $f \circ f^{-1} = \text{Id}_B$ .

**Exemple 38.** La rotation  $R(x, y) = (-y, x)$  de  $90^\circ$  en sens antihoraire est une bijection. Sa réciproque est la fonction  $R^{-1}(x, y) = (y, -x)$ . En effet,

$$(R \circ R^{-1})(x, y) = R(R^{-1}(x, y)) = R(y, -x) = (x, y).$$

**Exemple 39.** Calculons la réciproque de  $f(x) = 2^{x-3} + 4$ . On pose  $y = 2^{x-3} + 4$  et on isole  $x$ . On obtient

$$\begin{aligned} y = 2^{x-3} + 4 &\Rightarrow y - 4 = 2^{x-3} \\ &\Rightarrow \log_2(y - 4) = \log_2(2^{x-3}) \\ &\Rightarrow \log_2(y - 4) = x - 3 \\ &\Rightarrow \log_2(y - 4) + 3 = x. \end{aligned}$$

Ainsi,  $f^{-1}(x) = \log_2(x - 4) + 3$ .



# Chapitre 4

## Suites et sommes

Autant en mathématiques qu'en informatique, les suites jouent un rôle fondamental. Les tableaux statiques en informatique peuvent être vus comme des suites finies.

**Définition 26.** Soit  $I \subseteq \mathbb{Z}$  et  $S$  un ensemble quelconque. Une *suite* de  $S$  est une fonction  $a : I \rightarrow S$ . Les éléments de  $I$  sont appelés *indices* de la suite  $a$ .

Souvent,  $I = \mathbb{N}$  ou  $I = \{1, 2, \dots, n\}$  ou encore  $I = \{0, 1, 2, \dots, n-1\}$ . De plus, pour alléger la notation, on dénote la suite par  $\{a_i\}_{i \in I}$  et, plutôt que d'utiliser la notation fonctionnelle  $a(i)$  pour dénoter l'élément d'indice  $i$ , on écrit  $a_i$ . On dit que  $a_i$  est le *terme général* de la suite.

### 4.1 Suites numériques

Bien qu'on puisse construire des suites à partir de n'importe quel ensemble, on s'intéresse souvent aux suites numériques.

**Exemple 40.** Soit  $\{a_i\}_{i \in \mathbb{N}}$  définie par  $a_i = 2^i$ . Alors les neuf premiers termes de la suite sont

$$1, 2, 4, 8, 16, 32, 64, 128, 256.$$

Il s'agit de la suite des *puissances de 2*.

**Exemple 41.** Soit  $\{a_i\}_{i \in \mathbb{N}}$  définie par  $a_i = (-1)^i$ . Alors

$$\begin{aligned} a_0 &= 1 \\ a_1 &= -1 \\ a_2 &= 1 \\ a_3 &= -1 \\ &\dots \end{aligned}$$

Certains types de suites numériques apparaissent régulièrement en informatique.

**Définition 27.** Soit  $r$  un réel positif et  $b$  un réel quelconque. Alors la suite  $\{a_i\}_{i \in \mathbb{N}}$  définie par  $a_i = ir + b$  est appelée *suite arithmétique* de *raison*  $r$  et de *terme initial*  $b$ .

**Exemple 42.** La suite  $\{a_i\}_{i \in \mathbb{N}}$  définie par  $a_i = 4i + 1$  est une suite arithmétique de raison 4 et de terme initial 1. Ses premiers termes sont

$$1, 5, 9, 13, 17, \dots$$

**Définition 28.** Soit  $r$  un réel positif et  $b$  un réel quelconque. Alors la suite  $\{a_i\}_{i \in \mathbb{N}}$  définie par  $a_i = br^i$  est appelée *suite géométrique* de *raison*  $r$  et de *terme initial*  $b$ .

**Exemple 43.** La suite

- $1, 3, 9, 27, 81, 243, \dots$  est une suite géométrique de raison 3 et de terme initial 1;
- $1, 1/2, 1/4, 1/8, \dots$  est une suite géométrique de raison  $1/2$  et de terme initial 1.

## 4.2 Suites récursives

Il est très fréquent de manipuler des suites qui sont définies de façon récursives. L'un des plus connues est sans doute la suite  $\{f_i\}_{i \in \mathbb{N}}$  de Fibonacci, qu'on définit par

$$f_0 = 1, \quad f_1 = 1 \quad \text{et} \quad f_i = f_{i-1} + f_{i-2}, \text{ pour } n \geq 2.$$

Ses premiers termes sont

$$\begin{aligned} f_0 &= 1 \\ f_1 &= 1 \\ f_2 &= f_1 + f_0 = 1 + 1 = 2 \\ f_3 &= f_2 + f_1 = 2 + 1 = 3 \\ f_4 &= f_3 + f_2 = 3 + 2 = 5 \\ f_5 &= f_4 + f_3 = 5 + 3 = 8 \\ f_6 &= f_5 + f_4 = 8 + 5 = 13 \\ &\dots \end{aligned}$$

La suite des *rapport successifs*  $\{r_i\}_{i \in \mathbb{N}}$  de la suite de Fibonacci est définie par  $r_i = f_{i+1}/f_i$ . Ses premiers termes sont

$$\begin{aligned} r_0 &= \frac{f_1}{f_0} = 1 \\ r_1 &= \frac{f_2}{f_1} = 2 \\ r_2 &= \frac{f_3}{f_2} = \frac{3}{2} = 1.5 \\ r_3 &= \frac{f_4}{f_3} = \frac{5}{3} \approx 1.6667 \\ r_4 &= \frac{f_5}{f_4} = \frac{8}{5} \approx 1.6 \\ r_5 &= \frac{f_6}{f_5} = \frac{13}{8} = 1.625 \\ r_6 &= \frac{f_7}{f_6} = \frac{21}{13} \approx 1.6154 \end{aligned}$$

On peut montrer qu'elle converge vers  $1.618034 \dots$ . Ce nombre est appelé le *nombre d'or*.

**Exemple 44.** Considérons la suite indicée par  $\mathbb{N}$  définie par

$$a_0 = 1, \quad a_1 = 2 \quad a_i = a_{i-1} - a_{i-2}.$$

Alors les premiers termes sont

$$\begin{aligned} a_0 &= 1 \\ a_1 &= 2 \\ a_2 &= a_1 - a_0 = 2 - 1 = 1 \\ a_3 &= a_2 - a_1 = 1 - 2 = -1 \\ a_4 &= a_3 - a_2 = -1 - 1 = -2 \\ a_5 &= a_4 - a_3 = -2 - (-1) = -1 \\ a_6 &= a_5 - a_4 = -1 - (-2) = 1 \\ a_7 &= a_6 - a_5 = 1 - (-1) = 2 \\ &\dots \end{aligned}$$

Nous terminons en discutant d'une conjecture dont l'énoncé est très simple. Considérons la fonction  $f : \mathbb{N} \rightarrow \mathbb{N}$  définie par

$$f(n) = \begin{cases} n/2, & \text{si } n \text{ est pair;} \\ 3n + 1, & \text{si } n \text{ est impair.} \end{cases}$$

Alors on définit la *suite de Syracuse*  $\{a_i\}_{i \in \mathbb{N}}$  d'un nombre  $n$  par

$$a_i = \begin{cases} n, & \text{si } i = 0; \\ f(a_{i-1}), & \text{si } i > 0. \end{cases}$$

Par exemple, prenons  $n = 6$ . Alors les premiers termes de la suite de Syracuse de  $n$  sont

$$6, 3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, \dots$$

**Conjecture 1.** Soit  $n$  un nombre naturel non nul. Alors la suite de Syracuse de  $n$  converge vers le cycle  $(1, 4, 2)$ .

## 4.3 Mots

Il est intéressant de considérer des suites non numériques. Un type de suite particulièrement important est celui de mots ou chaînes de caractères.

**Définition 29.** Un *alphabet* est un ensemble fini  $A$  dont les éléments sont appelés *lettres*. On dit que  $w$  est un *mot* sur l'alphabet  $A$  si c'est une suite de lettres prises dans  $A$ . On écrit  $w = w_1 w_2 w_3 \dots w_n$  plutôt que  $\{w_i\}_{i \in \{1, 2, \dots, n\}}$  pour alléger la notation.

La *longueur* d'un mot  $w$  est le nombre de lettres est dénotée  $|w|$ . Le *mot vide* est l'unique mot de longueur 0 et il est noté  $\varepsilon$ .

**Exemple 45.** Soit  $A = \{0, 1\}$  l'alphabet binaire. Alors  $w = 01011101$  est un mot de longueur 8 sur  $A$ . On écrit donc  $|w| = 8$ .

L'opération de base sur les mots est la concaténation. Soient  $u = u_1u_2u_3 \cdots u_m$  et  $v = v_1v_2v_3 \cdots v_m$  deux mots. Alors la concaténation de  $u$  et de  $v$ , notée  $uv$  ou  $u \cdot v$ , est le mot

$$uv = u_1u_2u_3 \cdots u_mv_1v_2 \cdots v_n.$$

**Exemple 46.** Si  $u = 01011$  et  $v = 010$ , alors  $uv = 01011010$  et  $vu = 01001011$ . On en conclut que la concaténation n'est pas commutative.

Les opérations logiques s'étendent naturellement aux chaînes binaires :

**Exemple 47.** Soient  $u = 01011$  et  $v = 10010$ . Alors

$$\begin{aligned} u \wedge v &= 01011 \wedge 10010 \\ &= 00010, \\ u \vee v &= 01011 \vee 10010 \\ &= 11011, \\ u \oplus v &= 01011 \oplus 10010 \\ &= 11001. \end{aligned}$$

L'opérateur logique de négation est appelé *complément* :

$$\overline{01011} = 10100.$$

Une dernière opération pratique est celle de l'image miroir.

**Définition 30.** Soit  $w = w_1w_2w_3 \cdots w_{n-1}w_n$  un mot quelconque. Alors l'*image miroir* de  $w$ , notée  $\tilde{w}$  est le mot

$$\tilde{w} = w_nw_{n-1} \cdots w_3w_2w_1.$$

Un mot qui vérifie l'équation  $w = \tilde{w}$  est appelé *palindrome*.

**Exemple 48.** Soit  $w = abaab$ . Alors  $\tilde{w} = baaba$ . Ce n'est donc pas un palindrome. Par contre, le mot  $u = \clubsuit \diamond \heartsuit \heartsuit \diamond \clubsuit$  est un palindrome.

## 4.4 Sommes

Lorsqu'on manipule des suites numériques, on doit souvent calculer leur somme. On peut même évaluer certaines sommes infinies !

**Définition 31.** Soit  $\{a_i\}_{i \in I}$  une suite de nombres. Alors sa *somme* est donnée par

$$\sum_{i \in I} a_i.$$

Si  $I = \{m, m+1, \dots, n-1, n\}$ , où  $m$  et  $n$  sont deux entiers,  $m < n$ , alors on écrit plutôt

$$\sum_{i=m}^n a_i = a_m + a_{m+1} + \dots + a_{n-1} + a_n.$$

Souvent,  $I = \{0, 1, 2, \dots, n-1\}$  ou  $I = \{1, 2, 3, \dots, n\}$  et alors on obtient une somme de la forme

$$\sum_{i=0}^{n-1} a_i \quad \text{ou} \quad \sum_{i=1}^n a_i.$$

**Exemple 49.** On a

$$\begin{aligned} \sum_{i=0}^3 (-1)^i 2^i &= (-1)^0 2^0 + (-1)^1 2^1 + (-1)^2 2^2 + (-1)^3 2^3 \\ &= 1 \cdot 1 - 1 \cdot 2 + 1 \cdot 4 - 1 \cdot 8 \\ &= 1 - 2 + 4 - 8 \\ &= -5. \end{aligned}$$

Supposons que  $I = \{2, 3, 5, 7, 11\}$ . Alors

$$\sum_{i \in I} i = 2 + 3 + 5 + 7 + 11 = 28.$$

Aussi,

$$\sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}.$$

Cette dernière somme est appelé *somme harmonique* et apparaît dans plusieurs analyses d'algorithmes.

Une somme qu'on rencontre fréquemment est celle des  $n$  premiers entiers. Par exemple, la somme des 8 premiers entiers est

$$\sum_{i=1}^8 i = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36.$$

On aimerait avoir une formule dans le cas général.

**Théorème 1.** Soit  $n$  un entier positif. Alors

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}.$$

**Démonstration.** Posons

$$S = 1 + 2 + 3 + \dots + (n-2) + (n-1) + n.$$

En écrivant les termes dans l'ordre inverse, on a donc

$$S = n + (n-1) + (n-2) + \dots + 3 + 2 + 1.$$

On constate donc que

$$\begin{aligned} 2S &= (n+1) + (n+1) + (n+1) + \dots + (n+1) + (n+1) + (n+1) \\ &= n(n+1), \end{aligned}$$

ce qui entraîne  $S = n(n+1)/2$ .

Lorsqu'on manipule des sommes, certaines propriétés sont particulièrement utiles :

$$\begin{aligned}\sum_{i \in I} (a_i \pm b_i) &= \sum_{i \in I} a_i \pm \sum_{i \in I} b_i \\ \sum_{i \in I} ca_i &= c \sum_{i \in I} a_i \\ \sum_{i \in I} c &= |I|c.\end{aligned}$$

Ces propriétés nous permettent entre autres de calculer une formule pour la somme de n'importe quelle suite arithmétique :

**Théorème 2.** Soit  $r$  un réel positif et  $b$  un réel quelconque. Alors

$$\sum_{i=1}^n (ir + b) = \frac{rn(n+1)}{2} + nb.$$

**Démonstration.** On a

$$\sum_{i=1}^n (ir + b) = r \sum_{i=1}^n i + \sum_{i=1}^n b = \frac{rn(n+1)}{2} + nb.$$

Un dernier type de somme qui est souvent rencontré concerne les suites géométriques.

**Théorème 3.** Soit  $r$  un réel positif et  $b$  un réel quelconque. Alors

$$\sum_{i=0}^n br^i = \frac{b(1-r^{n+1})}{1-r}.$$

**Démonstration.** Posons

$$S = \sum_{i=0}^n br^i = b + br + br^2 + \dots + br^n.$$

Si on multiplie par  $r$ , on obtient

$$rS = br + br^2 + br^3 + \dots + br^n + br^{n+1}.$$

En soustrayant, on trouve

$$S - rS = b - br^{n+1}.$$

Il ne reste qu'à isoler  $S$ . Or

$$\begin{aligned}S - rS = b - br^{n+1} &\Rightarrow S(1-r) = b(1-r^{n+1}) \\ &\Rightarrow S = \frac{b(1-r^{n+1})}{1-r},\end{aligned}$$

tel que voulu.

# Chapitre 5

## Nombres entiers et division

Dans ce chapitre, nous nous intéressons à certaines propriétés fondamentales des nombres entiers.

### 5.1 Divisibilité

**Définition 32.** Soient  $a \in \mathbb{Z}^*$  et  $b \in \mathbb{Z}$ . On dit que  $a$  *divise*  $b$ , noté  $a \mid b$ , s'il existe  $c \in \mathbb{Z}$  tel que  $b = ac$ . Sinon, on écrit  $a \nmid b$ . De plus, on dit alors que  $a$  est un *facteur* de  $b$  et que  $b$  est un *multiple* de  $a$ .

**Exemple 50.**  $3 \mid 12$ , mais  $5 \nmid 12$ .

La relation de divisibilité vérifie certaines propriétés élémentaires.

**Proposition 3.** Soient  $a, b, c \in \mathbb{Z}$ .

- (i) Si  $a \mid b$  et  $a \mid c$ , alors  $a \mid (b + c)$ ;
- (ii) Si  $a \mid b$ , alors  $a \mid bk$  pour tout  $k \in \mathbb{Z}$ ;
- (iii) Si  $a \mid b$  et  $b \mid c$ , alors  $a \mid c$ .

**Démonstration.** (i) Supposons que  $a \mid b$  et  $a \mid c$ . Alors il existe  $m, n \in \mathbb{Z}$  tels que  $b = am$  et  $c = an$ . Par conséquent,  $b + c = am + an = a(m + n)$ . Comme  $m + n \in \mathbb{Z}$ , on en conclut que  $a \mid (b + c)$ .

(ii) En exercice.

(iii) Supposons que  $a \mid b$  et  $b \mid c$ . Alors il existe  $m, n \in \mathbb{Z}$  tels que  $b = am$  et  $c = bn$ . On a donc  $c = bn = (am)n = a(mn)$ . Or,  $mn \in \mathbb{Z}$  et donc  $a \mid c$ .

**Définition 33.** Soit  $n > 1$  un entier. Soit  $D(n)$  l'ensemble des diviseurs de  $n$ . Alors on dit que  $n$  est *premier* si  $D(n) = \{1, n\}$ . Sinon,  $n$  est dit *composé*.

Si on prend les entiers entre 2 et 20, alors ils se divisent comme suit :

Premiers	:	2, 3, 5, 7, 11, 13, 17, 19
Composés	:	4, 6, 8, 9, 12, 14, 15, 16, 18, 20

**Théorème 4** (Théorème fondamental de l'arithmétique). Tout entier positif  $n \geq 2$  s'écrit de façon unique comme un produit de nombres premiers (en ordre croissant).

**Exemple 51.** On a

$$\begin{aligned} 2 &= 2 \\ 15 &= 3 \cdot 5 \\ 48 &= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 = 2^4 \cdot 3 \\ 515 &= 5 \cdot 103. \end{aligned}$$

Il existe un test simple permettant de décider si un nombre est premier ou non.

**Théorème 5** (Critère de primalité). Si  $n$  est composé, alors  $i \mid n$ , pour un certain entier  $i$ ,  $2 \leq i \leq \sqrt{n}$ .

**Démonstration.** Soit  $n$  un entier composé. Alors il existe des entiers  $a$  et  $b$ ,  $a, b \geq 2$  et  $a, b \neq n$ , tels que  $n = ab$ . Supposons par contradiction que  $a, b > n$ . Alors  $ab > n \cdot n = n^2$ , ce qui est absurde. On en conclut que  $a \leq n$  ou  $b \leq n$ .

On a donc un algorithme simple pour vérifier si un nombre est premier ou non :

```

fonction ESTPREMIER( $n$  : entier  $\geq 2$ ) : booléen
     $i \leftarrow 2$ 
    tant que  $i \leq \sqrt{n} \wedge i \nmid n$  faire
         $i \leftarrow i + 1$ 
    fin tant que
    retourner  $i > n$ 
fin fonction

```

## 5.2 Arithmétique modulaire

En programmation, tout langage de programmation fournit généralement une opération de division entière et l'opération modulo. Par exemple, le programme Java qui suit

```

public class DivMod {
    public static void main(String[] args) {
        System.out.println("342□/□25□=□" + (342 / 25));
        System.out.println("342□%□25□=□" + (342 % 25));
    }
}

```

affiche à l'écran

```

13
17

```

Plus généralement, on a la relation suivante :

**Théorème 6.** Soit  $a, d$  deux entiers,  $d > 0$ . Alors il existe des entiers unique  $q$  et  $r$  tels que

$$a = dq + r, \quad 0 \leq r < d.$$

Les nombres  $a, d, q, r$  sont appelés respectivement le *dividende*, le *diviseur*, le *quotient* et le *reste*.

Attention aux valeurs négatives. Il suffit de garder en tête que le reste  $r$  vérifie les inégalités  $0 \leq r < d$  :

**Exemple 52.** Division  $-11$  par  $3$ . Alors

$$-11 = (-4) \cdot 3 + 1,$$

**Définition 34.** Soit  $a$  et  $m$  deux entiers,  $m > 0$ . Alors  $a \bmod m$  est le reste de la division de  $a$  par  $m$ .

**Exemple 53.**  $17 \bmod 5 = 2$ ,  $9 \bmod 3 = 0$ ,  $-13 \bmod 4 = 3$ .

On s'intéresse souvent aux nombres ayant le même reste par une division entière.

**Définition 35.** Soit  $a, b, m$  des entiers,  $m > 0$ . Alors on dit que  $a$  est congru à  $b$  modulo  $m$  si  $m \mid (b - a)$ . On écrit alors  $a \equiv b \pmod{m}$ .

**Proposition 4.** Soient  $a, b, m$  des entiers,  $m > 0$ . Alors  $a \equiv b \pmod{m}$  si et seulement si  $a \bmod m = b \bmod m$ .

**Exemple 54.** On a  $5 \equiv 17 \pmod{6}$ , puisque  $6 \mid (17 - 5)$ . De façon équivalente,  $5 \bmod 6 = 5 = 17 \bmod 6$ .

**Exemple 55.** L'ensemble des valeurs  $x$  telles que  $2 \equiv x \pmod{5}$  est

$$\{\dots, -13, -8, -3, 2, 7, 12, 17, \dots\} = 5\mathbb{Z} + 2.$$

Les propriétés suivantes découlent directement de la définition :

**Théorème 7.** Soit  $m > 0$  un entier.

- (i) Si  $a, b$  sont des entiers, alors  $a \equiv b \pmod{m}$  si et seulement s'il existe  $k \in \mathbb{Z}$  tel que  $a = km + b$ ;
- (ii) Si  $a, b, c, d$  sont des entiers tels que  $a \equiv b \pmod{m}$  et  $c \equiv d \pmod{m}$ , alors
  - (a)  $a + c \equiv b + d \pmod{m}$
  - (b)  $ac \equiv bd \pmod{m}$ .

**Démonstration.** (i) est laissé en exercice.

Démontrons (ii). Supposons que  $a \equiv b \pmod{m}$  et  $c \equiv d \pmod{m}$ . Par (i), il existe des entiers  $k_1, k_2$  tels que

$$a = k_1m + b \quad \text{et} \quad c = k_2m + d.$$

Alors

$$\begin{aligned} a + c &= (k_1m + b) + (k_2m + d) \\ &= (k_1m + k_2m) + (b + d) \\ &= (k_1 + k_2)m + (b + d) \\ ac &= (k_1m + b)(k_2m + d) \\ &= k_1k_2m^2 + k_1md + k_2mb + bd \\ &= (k_1k_2m + k_1d + k_2b)m + bd, \end{aligned}$$

ce qui entraîne  $a + c \equiv b + d \pmod{m}$  et  $ac \equiv bd \pmod{m}$ .

### 5.3 Les fonctions pgcd et ppcm

Une autre notion fondamentale en théorie des nombres est celle de plus grand commun diviseur et plus petit commun multiple.

**Définition 36.** Soient  $a, b \in \mathbb{Z}$ , pas tous deux nuls.

- Le *plus grand commun diviseur* de  $a$  et  $b$ , noté  $\text{pgcd}(a, b)$ , est le plus grand entier divisant à la fois  $a$  et  $b$ .
- Le *plus petit commun multiple* de  $a$  et  $b$ , noté  $\text{ppcm}(a, b)$ , est le plus petit entier qui est multiple à la fois  $a$  et  $b$ .

En particulier, si  $\text{pgcd}(a, b) = 1$ , alors on dit que  $a$  et  $b$  sont *premiers entre eux*.

**Exemple 56.** Calculons le pgcd et le ppcm de 24 et 36. Soit  $D(n)$  l'ensemble des diviseurs du nombre  $n$ . Alors

$$\begin{aligned} D(24) &= \{1, 2, 3, 4, 6, 8, 12, 24\} \\ D(36) &= \{1, 2, 3, 4, 6, 9, 12, 18, 36\}. \end{aligned}$$

On voit donc que  $\text{pgcd}(24, 36) = 12$ . Soit  $M(n)$  l'ensemble des multiples de  $n$ . Alors

$$\begin{aligned} M(24) &= \{24, 48, 72, 96, 120, \dots\} \\ M(36) &= \{36, 72, 108, 144, \dots\}. \end{aligned}$$

On en conclut que  $\text{ppcm}(24, 36) = 72$ .

Les fonction pgcd et ppcm sont liées par la relation suivante :

**Théorème 8.** Soit  $a$  et  $b$  deux entiers pas tous deux nuls. Alors

$$ab = \text{pgcd}(a, b)\text{ppcm}(a, b).$$

La décomposition en facteurs premiers permet également de calculer le pgcd et le ppcm de deux nombres.

**Exemple 57.** Calculons  $\text{pgcd}(54, 60)$  et  $\text{ppcm}(54, 60)$ . Notons que

$$\begin{aligned} 54 &= 6 \cdot 9 = 2 \cdot 3^4 = 2^1 \cdot 3^3 \cdot 5^0 \cdot 7^0 \\ 60 &= 4 \cdot 15 = 2^2 \cdot 3 \cdot 5 = 2^2 \cdot 3^1 \cdot 5^1 \cdot 7^0. \end{aligned}$$

Ainsi,

$$\begin{aligned} \text{pgcd}(54, 60) &= 2^1 \cdot 3^1 \cdot 5^0 \cdot 7^0 = 6 \\ \text{ppcm}(54, 60) &= 2^2 \cdot 3^3 \cdot 5^1 \cdot 7^0 = 540. \end{aligned}$$

## 5.4 Algorithme d'Euclide

Il existe un algorithme bien meilleur qui permet de calculer le plus grand commun diviseur de deux nombres, mentionné dans Éléments d'Euclide. Il est basé sur la proposition suivante :

**Proposition 5.** Soit  $a, b$  deux entiers,  $b > 0$ . Alors  $\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b)$ .

**Démonstration.** La division entière de  $a$  par  $b$  s'écrit

$$a = bq + r,$$

où  $q$  et  $r$  sont le quotient et le reste. Étant donné deux entiers  $x$  et  $y$ , dénotons par  $D(x, y)$  l'ensemble des diviseurs communs à  $x$  et à  $y$ . Nous montrons maintenant que  $D(a, b) = D(b, r)$ .

Soit  $d \in D(a, b)$ . Alors  $d \mid a$  et  $d \mid b$ . Aussi,  $d \mid bq$  et donc  $d \mid (bq - a)$ , c'est-à-dire que  $d \mid r$ . On en conclut que  $d \in D(b, r)$ .

Réciproquement, supposons que  $d \in D(b, r)$ . Alors  $d \mid b$  et  $d \mid r$ . On en déduit que  $d \mid bq$  et alors  $d \mid (bq + r)$ , ce qui entraîne que  $d \mid a$ . Ainsi,  $d \in D(a, b)$ .

Comme  $a, b$  et  $b, r$  partagent les mêmes diviseurs, ils ont donc le même plus grand diviseur.  $\square$

**Exemple 58.** On a

$$\begin{aligned} \text{pgcd}(75, 134) &= \text{pgcd}(134, 75 \bmod 134) \\ &= \text{pgcd}(134, 75) \\ &= \text{pgcd}(75, 134 \bmod 75) \\ &= \text{pgcd}(75, 59) \\ &= \text{pgcd}(59, 75 \bmod 59) \\ &= \text{pgcd}(59, 16) \\ &= \text{pgcd}(16, 59 \bmod 16) \\ &= \text{pgcd}(16, 11) \\ &= \text{pgcd}(11, 16 \bmod 11) \\ &= \text{pgcd}(11, 5) \\ &= \text{pgcd}(5, 11 \bmod 5) \\ &= \text{pgcd}(5, 1) \\ &= \text{pgcd}(1, 5 \bmod 1) \\ &= \text{pgcd}(1, 0) \\ &= 1. \end{aligned}$$

D'autre part,

$$\begin{aligned} \text{pgcd}(24, 36) &= \text{pgcd}(36, 24 \bmod 36) \\ &= \text{pgcd}(36, 24) \\ &= \text{pgcd}(24, 36 \bmod 24) \\ &= \text{pgcd}(24, 12) \\ &= \text{pgcd}(12, 24 \bmod 12) \\ &= \text{pgcd}(12, 0) \\ &= 12. \end{aligned}$$

Une autre application de l'algorithme d'Euclide est qu'il permet d'écrire un nombre dans n'importe quelle base.

**Exemple 59.** Écrivons 34 en base 2 (en binaire). On a

$$\begin{aligned} 34 &= 2 \cdot 17 + 0 \\ 17 &= 2 \cdot 8 + 1 \\ 8 &= 2 \cdot 4 + 0 \\ 4 &= 2 \cdot 2 + 0 \\ 2 &= 2 \cdot 1 + 0 \\ 1 &= 2 \cdot 0 + 1 \end{aligned}$$

On en conclut donc que  $(34)_2 = 100010$ . De la même façon, si on veut écrire 4017 en base 16 (hexadécimale), alors on obtient

$$\begin{aligned} 4017 &= 16 \cdot 251 + 1 \\ 251 &= 16 \cdot 15 + 11 \\ 15 &= 16 \cdot 0 + 15, \end{aligned}$$

et alors  $(4017)_{16} = FB1$ .

## 5.5 Cryptographie

En 1976, Rivest, Shamir et Adleman ont proposé un algorithme (nommé RSA) basé sur le petit théorème de Fermat.

**Théorème 9** (Fermat, 1640). Soit  $p$  un nombre premier et  $a \in \mathbb{Z}$ , tel que  $p \nmid a$ . Alors

$$a^{p-1} \equiv 1 \pmod{p}.$$

**Exemple 60.** Prenons  $p = 7$  et  $a = 10$ . Alors

$$a^{p-1} = 10^6.$$

Or,

$$\begin{aligned} 10 &\equiv 3 \pmod{7} \Rightarrow 10^2 \equiv 3^2 = 9 \equiv 2 \pmod{7} \\ &\Rightarrow 10^3 \equiv 3 \cdot 2 \equiv 6 \pmod{7} \\ &\Rightarrow 10^4 \equiv 3 \cdot 6 = 18 \equiv 4 \pmod{7} \\ &\Rightarrow 10^5 \equiv 3 \cdot 4 = 12 \equiv 5 \pmod{7} \\ &\Rightarrow 10^6 \equiv 3 \cdot 5 = 15 \equiv 1 \pmod{7}, \end{aligned}$$

c'est-à-dire que

$$a^{p-1} = 10^6 \equiv 1 \pmod{7},$$

ce qui confirme le petit théorème de Fermat.

Dans une version simplifiée de l'algorithme RSA, on procède comme suit :

1. On transforme tout d'abord le message en un entier  $M$  :

$$\text{fromage} \Rightarrow M = 05\ 17\ 14\ 12\ 00\ 06\ 04$$

2. On choisit ensuite deux nombres premiers très grands  $p$  et  $q$ , ainsi qu'un entier  $e$  tel que  $\text{pgcd}(e, (p-1)(q-1)) = 1$  :

$$\begin{aligned} p &= 43, \\ q &= 59, \\ n &= pq = 43 \cdot 59 = 2537 \text{ (clé publique)} \\ e &= 13 \text{ (} e = \text{encodage)}. \end{aligned}$$

On a bien  $\text{pgcd}(13, 42 \cdot 58) = 1$ .

3. Puis on code chaque bloc par  $M^e \bmod n$  :

$$\begin{aligned} 0517^{13} \bmod 2537 &= 1334 \\ 1412^{13} \bmod 2537 &= 0509 \\ 0006^{13} \bmod 2537 &= 2371 \\ 0426^{13} \bmod 2537 &= 0032 \end{aligned}$$

Le message codé est donc  $C = 1334\ 0509\ 2371\ 0032$ .

4. Pour décoder le message, il faut connaître la clé privée  $d$  qui est obtenue par rapport à  $e$  et  $n$  en vertu du petit théorème de Fermat. Dans l'exemple ci-haut,  $d = 937$  et on retrouve les blocs en calculant  $C^d \bmod n$  :

$$\begin{aligned} 1334^{937} \bmod 2537 &= 0517 \\ 0509^{937} \bmod 2537 &= 1412 \\ 2371^{937} \bmod 2537 &= 0006 \\ 0032^{937} \bmod 2537 &= 0426 \end{aligned}$$

On remarque en particulier que même si quelqu'un connaît le message codé  $C$  et la clé publique  $n$ , alors il ne peut retrouver  $M$ , sauf qu'il connaît  $p$  et  $q$ .



# Chapitre 6

## Démonstrations

Nous abordons dans ce chapitre différentes techniques de démonstration utilisées en mathématiques. Dans la plupart des cas, celles-ci doivent être combinées afin d'obtenir un raisonnement complet.

### 6.1 Règles d'inférence

Nous décrivons maintenant les règles considérées comme valide dans un raisonnement mathématique. Tout d'abord, nous introduisons la notation suivante :

$$\frac{\begin{array}{c} h_1 \\ h_2 \\ h_3 \quad (\text{hypothèses}) \\ \vdots \\ h_n \end{array}}{\therefore c \quad (\text{conclusion})}$$

Sous forme logique, cette règle correspond à l'expression  $(h_1 \wedge h_2 \wedge h_3 \wedge \dots \wedge h_n) \rightarrow c$ .

La règle la plus utilisée est celle du *modus ponens* :

$$\frac{p \quad p \rightarrow q}{\therefore q} \quad \frac{n \text{ pair} \quad n \text{ pair} \rightarrow n^2 \text{ pair}}{\therefore n^2 \text{ pair}}$$

Ensuite, il y a le *modus tollens* :

$$\frac{\neg q \quad p \rightarrow q}{\therefore \neg p} \quad \frac{n^2 \text{ impair} \quad n \text{ pair} \rightarrow n^2 \text{ pair}}{\therefore n^2 \text{ impair}}$$

La *simplification* :

$$\frac{p \wedge q}{\therefore p} \quad \frac{n \text{ est pair et } n \leq 10}{\therefore n \text{ est pair}}$$

Le *syllogisme disjonctif* :

$$\frac{p \vee q \quad \neg p}{\therefore q} \quad \frac{n \text{ est pair ou } n \text{ est impair} \quad n \text{ n'est pas pair}}{\therefore n \text{ est impair}}$$

Le *sylogisme par hypothèse* :

$$\frac{p \rightarrow q}{q \rightarrow r} \quad \frac{n \text{ est pair} \rightarrow 2 \mid n}{2 \mid n \rightarrow n = 2k, \text{ pour un certain } k \in \mathbb{Z}}$$

$$\therefore p \rightarrow r \quad \therefore n \text{ est pair} \rightarrow n = 2k, \text{ pour un certain } k \in \mathbb{Z}$$

Souvent, les démonstrations mathématiques sont longues et sont décomposées en “blocs” :

- Les *axiomes* sont des énoncés qu’on suppose vrais et qu’on accepte sans démonstration;
- Les *définitions* introduisent des termes, généralement dans le but de rendre le raisonnement plus simple et plus compréhensible;
- Les *propositions* sont des énoncés vrais qu’on peut déduire des définitions;
- Les *lemmes* sont des énoncés vrais dont l’objectif principal est de démontrer un théorème (un résultat jugé plus important);
- Les *théorèmes* sont des énoncés vrais et jugés importants;
- Les *corollaire* sont des conséquences plus ou moins immédiates des théorèmes.

## 6.2 Démonstration directe

La démonstration directe est le raisonnement de base de toute démonstration mathématique. On part d’une hypothèse et on obtient la conclusion par une suite d’inférence.

**Exemple 61.** Montrer que  $A \oplus B \subseteq A \cup B$ , peu importe les ensembles  $A$  et  $B$ .

Supposons que  $x \in A \oplus B$ . Alors  $x \in A$  ou  $x \in B$ , mais  $x \notin A \cap B$ . En particulier,  $x \in A$  ou  $x \in B$ , c’est-à-dire  $x \in A \cup B$ , tel que voulu.

**Exemple 62.** Montrer que le carré d’un nombre impair est impair.

Supposons que  $n$  est impair. Alors il existe  $k \in \mathbb{Z}$  tel que  $n = 2k + 1$ . Or,

$$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1.$$

On en déduit donc que  $n^2$  est impair.

Parfois, même si la démonstration est directe, il n’est pas évident de trouver l’argument :

**Exemple 63.** Démontrez que pour tous  $a, b, c \in \mathbb{R}_+^*$ , si  $a + b \leq c$ , alors

$$\log_2 a + \log_2 b \leq 2 \log_2 c - 2.$$

On a

$$\begin{aligned} a + b \leq c &\Rightarrow (a + b)^2 \leq c^2 \\ &\Rightarrow a^2 + 2ab + b^2 \leq c^2 \\ &\Rightarrow 2ab \leq c^2 \\ &\Rightarrow ab \leq \frac{c^2}{2}. \end{aligned}$$

En prenant le logarithme de chaque membre de l’inégalité, on obtient

$$\log_2(ab) \leq \log_2\left(\frac{c^2}{2}\right) \Rightarrow \log_2 a + \log_2 b \leq 2 \log_2 c - 2.$$

## 6.3 Contraposée

Dans la démonstration par contraposée, on démontre une implication  $p \rightarrow q$  en démontrant plutôt que  $\neg q \rightarrow \neg p$ .

**Exemple 64.** Montrer que si  $3n + 2$  est impair, alors  $n$  est impair.

Il est difficile de montrer ce résultat directement. On démontre plutôt la contraposée, c'est-à-dire qu'on suppose que  $n$  est pair. Alors il existe  $k \in \mathbb{Z}$  tel que  $n = 2k$ . Or,

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

ce qui entraîne que  $3n + 2$  est pair, et la démonstration est terminée.

**Exemple 65.** Rappelons qu'une fonction  $f : A \rightarrow B$  est injective si pour tout  $x, y \in A$ , on a que  $x \neq y$  implique  $f(x) \neq f(y)$ . Il est généralement plus facile de démontrer la contraposée de cette affirmation, c'est-à-dire de montrer que  $f(x) = f(y)$  implique  $x = y$ . Par exemple, montrons que  $f : \{0, 1, \dots, 99\} \rightarrow \{0, 1, \dots, 99\}$  définie par

$$f(x) = 3x \bmod 100$$

est injective.

Supposons que  $f(x) = f(y)$  pour  $x, y \in \{0, 1, \dots, 99\}$ . Alors

$$\begin{aligned} 3x \bmod 100 = 3y \bmod 100 &\Rightarrow 3x \equiv 3y \pmod{100} \\ &\Rightarrow 67 \cdot 3x \equiv 67 \cdot 3y \pmod{100} \\ &\Rightarrow 201x \equiv 201y \pmod{100} \\ &\Rightarrow x \equiv y \pmod{100} \\ &\Rightarrow y - x = 100k, \text{ pour un certain } k \in \mathbb{Z}. \end{aligned}$$

Or,  $0 \leq x, y \leq 99$ , ce qui entraîne  $-99 \leq y - x \leq 99$  et donc  $k = 0$ . On en conclut que  $y - x = 100 \cdot 0 = 0$ , c'est-à-dire  $x = y$ .

## 6.4 Démonstration par contradiction

Dans certains cas, il est plus facile de faire une démonstration en supposant que la conclusion est fausse et en déduisant une contradiction. Plus formellement, on peut démontrer  $p \rightarrow q$  en montrant que  $\neg q \rightarrow \text{faux}$ .

**Exemple 66.** Montrer qu'il existe une infinité de nombres premiers.

Supposons par contradiction qu'il en existe un nombre fini  $n$  et soient  $p_1, p_2, \dots, p_n$  tous ces nombres premiers. Posons  $p = p_1 p_2 \cdots p_n + 1$ . Tout d'abord, remarquons que  $p_1 \nmid p$ . En effet, si c'était le cas, on aurait  $p_1 \mid p - p_1 p_2 \cdots p_n = 1$ , ce qui est impossible, puisque le seul nombre qui divise 1 est 1, qui n'est pas premier. De la même façon, on a que  $p_2 \nmid p, p_3 \nmid p, \dots, p_n \nmid p$ . Par le théorème fondamental de l'arithmétique, on en déduit que  $p$  est premier, puisqu'aucun autre nombre premier ne le divise. Or,  $p > p_i$  pour  $i = 1, 2, \dots, n$ , c'est-à-dire que  $p$  n'est pas un nombre premier parmi  $p_1, p_2, \dots, p_n$ , ce qui est absurde. On en conclut donc qu'il doit exister une infinité de nombres premiers.

**Exemple 67.** Montrer que  $\sqrt{2}$  est irrationnel.

Juste avant de montrer le résultat, nous avons besoin de montrer le petit lemme suivant :

**Lemme 1.** Soit  $n \in \mathbb{Z}$ . Alors  $n$  est pair si et seulement si  $n^2$  est pair.

**Démonstration.** Montrons que  $n$  pair  $\rightarrow n^2$  pair. Si  $n$  est pair, alors il existe  $k \in \mathbb{Z}$  tel que  $n = 2k$ . Alors  $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$ , ce qui montre que  $n^2$  est pair.

Réciproquement, montrons que  $n^2$  pair  $\rightarrow n$  pair. Il est plus facile de montrer la contraposée, c'est-à-dire de montrer que  $n$  impair  $\rightarrow n^2$  est impair, ce qui a été à la section 6.2.

Par l'absurde, supposons au contraire que  $\sqrt{2}$  est rationnel. Alors il existe  $a, b \in \mathbb{Z}$ ,  $b \neq 0$ , tels que

$$\sqrt{2} = \frac{a}{b}.$$

Comme chaque rationnel peut être exprimé à l'aide d'une fraction réduite, on peut même choisir  $a$  et  $b$  de sorte que  $\text{pgcd}(a, b) = 1$ .

Or,

$$\sqrt{2} = \frac{a}{b} \Rightarrow 2 = \frac{a^2}{b^2} \Rightarrow a^2 = 2b^2.$$

On en déduit donc que  $a^2$  est pair. Par le lemme, ceci entraîne que  $a$  est pair. Il existe donc  $k \in \mathbb{Z}$  tel que  $a = 2k$  et alors

$$\begin{aligned} a^2 = 2b^2 &\Rightarrow (2k)^2 = 2b^2 \\ &\Rightarrow 4k^2 = 2b^2 \\ &\Rightarrow 2k^2 = b^2, \end{aligned}$$

et donc  $b^2$  est pair, de sorte que  $b$  aussi est pair. On a donc que  $a$  et  $b$  sont tous deux pairs, ce qui implique  $\text{pgcd}(a, b) \geq 2$ , contredisant  $\text{pgcd}(a, b) = 1$ . Ainsi,  $\sqrt{2}$  ne peut être rationnel (il ne peut être représenté par une fraction réduite).

## 6.5 Cas par cas

Dans certaines situations où il est difficile de démontrer directement un énoncé de la forme  $p \rightarrow q$ , il est parfois intéressant d'exprimer  $p$  comme une disjonction. Par exemple, si  $p \Leftrightarrow p_1 \cup p_2$ , alors on peut démontrer  $p_1 \rightarrow q$  et  $p_2 \rightarrow q$  plutôt que  $p \rightarrow q$ .

**Exemple 68.** Démontrez que  $n^3 - n$  est pair pour tout entier  $n$ .

Il est difficile de démontrer directement cette affirmation, mais cela devient beaucoup plus facile si on sépare en deux cas :

(1) Supposons que  $n$  est pair. Alors il existe  $k \in \mathbb{Z}$  tel que  $n = 2k$ . Donc

$$n^3 - n = (2k)^3 - 2k = 8k^3 - 2k = 2(4k^3 - k),$$

ce qui montre que  $n^3 - n$  est pair.

(2) Supposons maintenant que  $n$  est impair. Alors il existe  $k \in \mathbb{Z}$  tel que  $n = 2k + 1$ . Donc

$$\begin{aligned} n^3 - n &= (2k + 1)^3 - (2k + 1) \\ &= (4k^3 + 4k + 1)(2k + 1) - (2k + 1) \\ &= (4k^3 + 4k + 1 - 1)(2k + 1) \\ &= (4k^3 + 4k)(2k + 1) \\ &= 2(2k^3 + 2k)(2k + 1), \end{aligned}$$

ce qui montre que  $n^3 - n$  est pair dans ce cas aussi.

Il peut parfois y avoir plus de deux cas.

**Exemple 69.** Démontrez que pour toute paire de nombres réels  $x$  et  $y$ ,

$$\max(x, y) = \frac{x + y + |x - y|}{2}.$$

On distingue deux cas, selon que  $x - y \geq 0$  ou  $x - y < 0$ .

(1) Supposons que  $x - y \geq 0$ . Alors  $x \geq y$  et donc

$$\frac{x + y + |x - y|}{2} = \frac{x + y + x - y}{2} = \frac{2x}{2} = x = \max(x, y).$$

(2) Supposons maintenant que  $x - y < 0$ . Alors  $x < y$  et donc

$$\frac{x + y + |x - y|}{2} = \frac{x + y + -(x - y)}{2} = \frac{x + y - x + y}{2} = \frac{2y}{2} = y = \max(x, y).$$

## 6.6 Équivalences

Dans certains cas, on souhaite démontrer que plusieurs énoncés sont équivalents :

1.  $A \subseteq B$
2.  $A \cap \overline{B} = \emptyset$
3.  $\overline{A} \cup B = U$ .

Plutôt que de démontrer toutes les équivalences, on peut utiliser un schéma circulaire. Il suffit alors de démontrer les implications  $(1) \rightarrow (2)$ ,  $(2) \rightarrow (3)$  et  $(3) \rightarrow (1)$ .

Montrons  $(1) \rightarrow (2)$ . Supposons donc que  $A \subseteq B$  et montrons que  $A \cap \overline{B} = \emptyset$ . Par contradiction, supposons qu'il existe  $x \in A \cap \overline{B}$ . Alors en particulier,  $x \in A$  et  $x \in \overline{B}$ . Or, comme  $A \subseteq B$  et que  $x \in A$ , on a  $x \in B$ , ce qui est absurde, car on ne peut avoir à la fois  $x \in B$  et  $x \in \overline{B}$ .

Montrons maintenant que  $(2) \rightarrow (3)$ . Supposons que  $A \cap \overline{B} = \emptyset$ . Alors  $\overline{A \cap \overline{B}} = \overline{\emptyset}$  et alors  $\overline{A} \cup B = U$ , par la loi de Morgan.

Finalement, montrons que  $(3) \rightarrow (1)$ . On suppose donc que  $\overline{A} \cup B = U$ . Soit  $x \in A$ . Alors  $x \in U = \overline{A} \cup B$ , c'est-à-dire que  $x \in \overline{A}$  ou  $x \in B$ . Or, on sait que  $x \in A$ , c'est-à-dire qu'on a nécessairement  $x \in B$ . On a montré que  $x \in A$  implique  $x \in B$ , ce qui entraîne que  $A \subseteq B$ .



# Chapitre 7

## Algorithmes

En informatique, un problème est généralement résolu à l'aide d'un algorithme.

- On a un *problème* qu'on souhaite résoudre;
- On doit d'abord *modéliser* ce problème à l'aide d'un formalisme adéquat (logique, ensembles, suites, fonctions, matrices, graphes, etc.)
- On propose ensuite un *algorithme*;
- On doit s'assurer que la *solution* est générale.

### 7.1 Généralités

**Définition 37.** Un *algorithme* est une suite finie d'étapes permettant de résoudre un problème de façon générale.

**Exemple 70.** Étant donné un tableau (ou une liste) d'entiers, trouver le plus grand élément.

Une stratégie possible est :

1. On choisit le premier élément du tableau comme étant le maximum provisoire (initialisation);
2. On parcourt le reste du tableau en comparant successivement le maximum provisoire avec l'élément courant de la suite. Si ce dernier est plus grand, alors on met à jour le maximum provisoire, sinon, on poursuit.

Voici la trace de l'algorithme qu'on vient de décrire sur le tableau suivant :

$i$	0	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	15	17	4	-15	20	7	-11	20	0	74	25	62
max	15	17	17	17	20	20	20	20	20	74	74	74

Évidemment, la description en français comme on l'a fait plus haut est peu pratique, car il n'est pas toujours évident de passer de cette description à un langage informatique. D'autre part, la description en langage informatique n'est pas idéale non plus car :

- Il existe beaucoup de langages, et beaucoup de paradigmes;
- Le cycle de vie d'un langage de programmation est très court;
- Il y a beaucoup de particularités syntaxiques propres à chaque langage qui masque l'essentiel du message.

**Définition 38.** Le *pseudocode* est une forme intermédiaire d'expression à mi-chemin entre le français et les langages informatiques.

**Exemple 71.** Revenons au problème de trouver le maximum d'un tableau. Alors en pseudocode, on obtient quelque chose de la forme suivante :

```

1: fonction MAXIMUM( $T$  : tableau d'entiers) : entier
2:    $max \leftarrow T[0]$ 
3:   pour  $i = 1, 2, \dots, T.LONGUEUR - 1$  faire
4:     si  $T[i] > max$  alors
5:        $max \leftarrow T[i]$ 
6:     fin si
7:   fin pour
8:   retourner  $max$ 
9: fin fonction

```

## 7.2 Pseudocode

On doit retrouver les éléments suivants lorsqu'on conçoit un algorithme :

1. L'*entrée* : le type d'instance considérée dans le problème;
2. La *sortie* : le résultat produit par l'algorithme;
3. La *précision* : chaque étape doit être définie avec précision;
4. La *finitude* : l'algorithme doit toujours terminer;
5. L'*efficacité* : idéalement, on doit pouvoir mesurer le temps et l'espace utilisé par l'algorithme;
6. La *généralité* : toute instance doit être prise en compte.

De plus, lorsque vous écrivez du pseudocode, vous devez respecter les contraintes suivantes :

- Le pseudocode doit être écrit *uniquement* en français;
- Donnez des noms *significatifs* aux fonctions et aux variables;
- Votre code doit être *indenté uniformément*;
- N'utilisez pas la syntaxe d'un langage de programmation (Java ou autre). Par exemple, pas de `i++` ou de `--i`, de **switch**, etc.;
- Vous pouvez utiliser des expressions mathématiques, car c'est un langage *universel*;
- Les affectations sont identifiées par le symbole  $\leftarrow$ ;
- Identifiez bien les mots clés ou mots réservés (gras, souligné, etc.);
- Une *fonction* retourne toujours un résultat, alors qu'une *procédure* n'en retourne pas.
- N'hésitez pas au besoin à décomposer votre algorithme en *plusieurs* fonctions ou procédures.

## 7.3 Le problème de la fouille

Un des problèmes les plus importants est sans doute celui de la fouille, qui consiste à trouver un élément dans un tableau, une liste ou un ensemble.

Dans le cas d'un tableau d'entiers, on souhaite par exemple trouver le premier indice auquel se trouve un entier donné. Si l'élément ne se trouve pas dans le tableau, on retourne  $-1$ .

```

1: fonction FOUILLELINÉAIRE( $T$  : tableau d'entiers,  $x$  : entier) : entier
2:    $i \leftarrow 0$ 
3:   tant que  $i < T.LONGUEUR$  et  $x \neq T[i]$  faire
4:      $i \leftarrow i + 1$ 
5:   fin tant que
6:   si  $i < T.LONGUEUR$  alors
7:     retourner  $i$ 
8:   sinon
9:     retourner  $-1$ 
10:  fin si
11: fin fonction

```

Une variante du pseudocode donné plus haut consiste à retourner *vrai* si l'élément apparaît dans le tableau, *faux* sinon :

```

1: fonction CONTIENT( $T$  : tableau d'entiers,  $x$  : entier) : booléen
2:    $i \leftarrow 0$ 
3:   tant que  $i < T.LONGUEUR$  et  $x \neq T[i]$  faire
4:      $i \leftarrow i + 1$ 
5:   fin tant que
6:   retourner  $i < T.LONGUEUR$ 
7: fin fonction

```

**Exemple 72.** Faisons la trace d'une fouille linéaire pour l'élément 8 dans le tableau

$i$	0	1	2	3	4	5	6	7
$T[i]$	9	4	5	2	1	3	2	5

On peut représenter la trace à l'aide d'un tableau également :

$i$	0	1	2	3	4	5	6	7	8
$T[i]$	9	4	5	2	1	3	2	5	
$i < T.LONGUEUR$	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>F</i>
$x \neq T[i]$	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	<i>V</i>	

Lorsqu'on termine la boucle, la condition  $i < T.LONGUEUR$  est fausse, donc on retourne  $-1$ . De la même façon, si on recherche 5 dans le tableau, on a plutôt la trace

$i$	0	1	2	3	4	5	6	7	8
$T[i]$	9	4	5	2	1	3	2	5	
$i < T.LONGUEUR$	<i>V</i>	<i>V</i>	<i>V</i>						
$x \neq T[i]$	<i>V</i>	<i>V</i>	<i>F</i>						

Lorsqu'on quitte la boucle, la condition  $i < T.LONGUEUR$  est vraie, donc on retourne 2.

Il y a moyen d'améliorer la stratégie lors d'une fouille si on maintient un tableau trié.

**Exemple 73.** Considérons le tableau

$i$	0	1	2	3	4	5	6	7
$T[i]$	1	2	2	3	4	5	5	9

et supposons qu'on recherche la valeur 8. Y a-t-il une meilleure stratégie que dans le cas d'une fouille linéaire ?

Oui ! On effectue une fouille binaire (aussi appelée *recherche dichotomique*). L'idée est simple : on inspecte l'élément au milieu du tableau (appelé *pivot*) et on le compare avec la valeur cherchée. Si l'élément cherché est plus petit que le pivot, alors on sait qu'on peut éliminer la moitié droite du tableau. De façon symétrique, si l'élément cherché est plus grand, on sait qu'on peut éliminer la moitié gauche du tableau.

On répète cette idée jusqu'à ce que la valeur soit trouvée ou que tout le tableau ait été éliminé.

En pratique, quand on implémente une fouille binaire, on n'élimine pas réellement des portions de tableau. On utilise plutôt deux indices-curseurs ( $i$  et  $j$  par exemple) qui délimitent la portion de tableau qui n'a pas encore été éliminée. On obtient alors l'algorithme suivant :

```

1: fonction FOUILLEBINAIRE( $T$  : tableau d'entiers,  $x$  : entier) : entier
2:    $i \leftarrow 0$ 
3:    $j \leftarrow T.LONGUEUR - 1$ 
4:    $m \leftarrow \lfloor (i + j)/2 \rfloor$ 
5:   tant que  $i \leq j$  et  $T[m] \neq x$  faire
6:     si  $x < T[m]$  alors
7:        $j \leftarrow m - 1$ 
8:     sinon
9:        $i \leftarrow m + 1$ 
10:    fin si
11:     $m \leftarrow \lfloor (i + j)/2 \rfloor$ 
12:  fin tant que
13:  si  $i \leq j$  alors
14:    retourner  $m$ 
15:  sinon
16:    retourner  $-1$ 
17:  fin si
18: fin fonction

```

Nous voyons au chapitre suivant que la fouille binaire est beaucoup plus efficace que la fouille linéaire.

## 7.4 Plus grand commun diviseur

Rappelons la proposition suivante vue précédemment :

**Proposition 6.** Soit  $a, b$  deux entiers,  $b > 0$ . Alors  $\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b)$ .

Derrière cette proposition se cache un algorithme très simple (on suppose que  $a$  et  $b$  ne sont pas tous deux nuls) :

```

1: fonction PGCD( $a, b$  : naturels) : naturel
2:    $y \leftarrow b$ 
3:    $x \leftarrow a$ 
4:   tant que  $y \neq 0$  faire
5:      $z \leftarrow y$ 
6:      $y \leftarrow x \bmod y$ 

```

```

7:       $x \leftarrow z$ 
8:  fin tant que
9:  retourner  $x$ 
10: fin fonction

```

Effectuons la trace pour  $a = 4$  et  $b = 6$ . On obtient

$y$	6	4	2	0
$x$	4	6	4	2
$y \neq 0$	V	V	V	F
$z$	6	4	2	

Lorsque la boucle est terminée (car  $y = 0$ ), alors on retourne  $x = 2$ .

## 7.5 Palindromes

Nous terminons ce chapitre en nous concentrant sur le problème de décider si une chaîne de caractères est un palindrome ou non. Rappelons qu'une chaîne  $w$  est un palindrome si  $\tilde{w} = w$ . Une solution est l'algorithme suivante :

```

1: fonction ESTPALINDROME( $w$  : chaîne de caractères) : booléen
2:    $i \leftarrow 0$ 
3:    $j \leftarrow w.\text{LONGUEUR} - 1$ 
4:   tant que  $i \leq j$  et  $w[i] = w[j]$  faire
5:      $i \leftarrow i + 1$ 
6:      $j \leftarrow j - 1$ 
7:   fin tant que
8:   retourner  $i > j$ 
9: fin fonction

```

Remarquons que si  $w = \varepsilon$ , alors l'algorithme retourne *vrai*, ce qui est correct puisque le mot vide est un palindrome. Faisons la trace avec  $w = \text{ressasser}$ . On obtient

$(i, j)$	(0, 8)	(1, 7)	(2, 6)	(3, 5)	(4, 4)	(5, 3)
$i \leq j$	V	V	V	V	V	F
$(w[i], w[j])$	(r, r)	(e, e)	(s, s)	(s, s)	(a, a)	
$w[i] = w[j]$	V	V	V	V	V	

En quittant la boucle, on a que la proposition  $i > j$  est vraie et donc on retourne *vrai*. En revanche, si on prend  $w = \text{blob}$ , alors la trace est plutôt

$(i, j)$	(0, 3)	(1, 2)
$i \leq j$	V	V
$(w[i], w[j])$	(b, b)	(l, o)
$w[i] = w[j]$	V	F

et on comme la proposition  $i > j$  est fausse, on retourne *faux*.



# Chapitre 8

## Complexité

Lorsque nous concevons des algorithmes, il est important de mesurer leur efficacité. Il existe des outils mathématiques permettant de le faire, parmi lesquels on compte la notation *grand-O*.

Avant de l'introduire formellement, nous revoyons certains problèmes du chapitre 7.

### 8.1 Calcul du maximum

Commençons d'abord avec l'algorithme qui permet de calculer l'élément maximal d'un tableau :

```
1: fonction MAXIMUM( $T$  : tableau d'entiers) : entier
2:    $max \leftarrow T[0]$ 
3:   pour  $i = 1, 2, \dots, T.LONGUEUR - 1$  faire
4:     si  $T[i] > max$  alors
5:        $max \leftarrow T[i]$ 
6:     fin si
7:   fin pour
8:   retourner  $max$ 
9: fin fonction
```

Analysons ligne par ligne le temps d'exécution de chaque instruction :

- L'affectation  $max \leftarrow T[0]$  s'effectue en temps constant  $c_1$ . Nous pouvons aussi inclure l'affectation  $i \leftarrow 1$  dans  $c_1$ .
- Dans la boucle **pour**, on doit vérifier à chaque tour la condition  $i < T.LONGUEUR$ , ainsi qu'incrémenter la valeur de  $i$ . Soit  $c_2$  le temps pris ar pour vérifier ces conditions;
- La condition  $T[i] > max$  est vérifiée en temps constant, de même que l'affectation  $max \leftarrow T[i]$ . Appelons ce temps  $c_3$ .
- Finalement, on retourne le maximum en temps constant  $c_4$ .

Le nombre de tours de boucle est  $n - 1$ , où  $n$  est la longueur du tableau  $T$ . Au total, le temps est au plus

$$T(n) = c_1 + (n - 1)(c_2 + c_3) + c_4 = an + b,$$

où  $a$  et  $b$  sont des constantes. On dit alors que la complexité est *linéaire*.

## 8.2 Palindromes

Reprenons le pseudocode de l'algorithme permettant de décider si une chaîne de caractères est un palindrome :

```

1: fonction ESTPALINDROME( $w$  : chaîne de caractères) : booléen
2:    $i \leftarrow 0$ 
3:    $j \leftarrow w.\text{LONGUEUR} - 1$ 
4:   tant que  $i \leq j$  et  $w[i] = w[j]$  faire
5:      $i \leftarrow i + 1$ 
6:      $j \leftarrow j - 1$ 
7:   fin tant que
8:   retourner  $i > j$ 
9: fin fonction

```

Nous allons calculer, ligne par ligne, le temps d'exécution. Soit  $n$  la longueur de la chaîne  $w$ .

- Les instructions  $i \leftarrow 0$  et  $j \leftarrow w.\text{LONGUEUR} - 1$  s'effectuent en temps constant, disons  $c_1$ ;
- La vérification de la conditions  $i \leq j$  se fait en temps constant  $c_2$ ;
- La vérification de la condition  $w[i] = w[j]$  s'effectue en temps constant  $c_3$ ;
- Les instructions  $i \leftarrow i + 1$  et  $j \leftarrow j - 1$  sont aussi en temps constant  $c_4$ ;
- Finalement, on retourne la valeur booléenne  $i > j$  en temps constant  $c_5$ .

On remarque également que la boucle des lignes 4–7 est répétée au plus  $\lfloor n/2 \rfloor$  fois. Ainsi, au total, le temps d'exécution est

$$T(n) = c_1 + (c_2 + c_3 + c_4)\lfloor n/2 \rfloor + c_5 \leq an + b,$$

où  $a$  et  $b$  sont des constantes. La complexité est linéaire ici aussi.

## 8.3 Fouille binaire

Nous avons aussi vu une stratégie permettant d'identifier rapidement si une valeur apparaît ou non dans un tableau trié :

```

1: fonction FOUILLEBINAIRE( $T$  : tableau d'entiers,  $x$  : entier) : entier
2:    $i \leftarrow 0$ 
3:    $j \leftarrow T.\text{LONGUEUR} - 1$ 
4:    $m \leftarrow \lfloor (i + j)/2 \rfloor$ 
5:   tant que  $i \leq j$  et  $T[m] \neq x$  faire
6:     si  $x < T[m]$  alors
7:        $j \leftarrow m - 1$ 
8:     sinon
9:        $i \leftarrow m + 1$ 
10:    fin si
11:     $m \leftarrow \lfloor (i + j)/2 \rfloor$ 
12:  fin tant que
13:  si  $i \leq j$  alors
14:    retourner  $m$ 
15:  sinon

```

```

16:     retourner -1
17:   fin si
18: fin fonction

```

Cet algorithme est plus compliqué à analyser ligne par ligne. Par contre, on peut donner l'idée générale en se restreignant au cas où la taille du tableau est  $n = 2^k$ , pour un certain entier positif  $k$ .

On remarque en effet qu'à chaque tour de boucle, la taille du tableau réduit environ de moitié. Ainsi, le nombre total de tours de boucle est environ  $k$ . Toutes les autres opérations s'effectuent en temps constant. On obtient donc une complexité de

$$T(n) = ak + b.$$

Or,  $k = \log_2(n)$ , ce qui entraîne

$$T(n) = a \log_2(n) + b.$$

On dit alors que la complexité est *logarithmique*.

## 8.4 Différentes complexités

Il existe différentes mesures d'efficacité d'un algorithme. Dans ce cours, nous ne les abordons pas, mais elles sont aussi très importantes.

- La complexité au *pire cas* est le temps maximum qui peut être pris par l'algorithme dans tous les cas de figures possibles. C'est celui que nous étudions principalement dans ce cours.
- La complexité *moyenne* est le temps moyen pris par l'algorithme pour terminer. Le calcul d'une complexité moyenne nécessite la théorie des probabilités et est lié à la notion d'*espérance mathématique*.
- La complexité au *meilleur cas* est le temps minimum pris par l'algorithme dans tous les cas de figures possibles.
- Il existe également une notion appelée *complexité amortie*. Elle dépasse le cadre de ce cours et est typiquement vue dans des cours de niveau maîtrise.
- Finalement, il existe la notion de *complexité spatiale*, qui permet de mesurer l'espace mémoire supplémentaire utilisé par l'algorithme.

## 8.5 Notation grand-O

Il est en général fastidieux (et même impossible) de calculer le temps pris par un algorithme ligne par ligne. En outre, il est préférable d'avoir une appréciation simple de son efficacité. Un formalisme qui résout ce problème est la notation *grand-O*. Par exemple, nous avons vu que si le temps de calcul est  $T(n) = an + b$  pour certaines constantes  $a$  et  $b$ , alors on dit que ce temps est *linéaire*. Nous montrons maintenant d'où vient cette terminologie.

**Définition 39.** Soit  $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$  des fonctions. Alors on dit que  $f(n)$  est dans  $\mathcal{O}(g(n))$  s'il existe des constantes  $C$  et  $k$  telles que

$$f(n) \leq Cg(n), \quad \text{dès que } n \geq k.$$

Souvent, pas abus de notation, on écrit  $f(n) = \mathcal{O}(g(n))$ . On dit aussi que  $f$  ne croît pas plus vite que  $g$  ou que  $f$  a le même comportement asymptotique que  $g$ .

**Remarque 1.** Généralement, la fonction  $g(n)$  doit être simple (par exemple,  $n$ ,  $n^2$ , etc.). De plus, les constantes  $C$  et  $k$  ne sont pas uniques.

**Exemple 74.** Montrons que  $f(n) = 2n^2 - 5n + 3$  est dans  $\mathcal{O}(n^2)$ . Nous avons

$$\begin{aligned} f(n) &= 2n^2 - 5n + 3 \\ &\leq 2n^2 + 0 + 3n^2 \\ &= 5n^2, \end{aligned}$$

en autant que  $n \geq 1$ . Il suffit donc de prendre  $C = 5$  et  $k = 1$  dans la définition, c'est-à-dire qu'on a

$$f(n) \leq 5n^2, \quad \text{dès que } n \geq 1.$$

On en conclut que  $f(n) = \mathcal{O}(n^2)$ .

**Exemple 75.** Montrons que  $f(n) = n^2$  n'est pas dans  $\mathcal{O}(n)$ . On procède par l'absurde, c'est-à-dire qu'on suppose qu'il existe des constantes  $C$  et  $k$  telles que  $f(n) \leq Cn$  pour tout  $n \geq k$ . Par conséquent,  $n^2 \leq Cn$ , ce qui entraîne  $n \leq C$  pour tout  $n \geq k$ , ce qui est impossible. On en conclut donc que  $n^2$  n'est pas  $\mathcal{O}(n)$ .

Si on revient aux trois algorithmes vus plus tôt, on a donc que la complexité au pire cas de l'algorithme

- qui calcule le maximum d'un tableau de longueur  $n$  est dans  $\mathcal{O}(n)$ ;
- qui vérifie si une chaîne de longueur  $n$  est un palindrome est dans  $\mathcal{O}(n)$ ;
- qui effectue une fouille binaire dans un tableau trié de longueur  $n$  est  $\mathcal{O}(\log n)$ .

Certaines probabilités additionnelles de la notation grand-O sont particulièrement utiles :

**Théorème 10.** Soient  $f_1, f_2, g_1, g_2$  des fonctions telles que  $f_1(n) = \mathcal{O}(g_1(n))$  et  $f_2(n) = \mathcal{O}(g_2(n))$ . Alors

1.  $f_1(n) + f_2(n) = \mathcal{O}(\max(g_1(n), g_2(n)))$ ;
2.  $f_1(n)f_2(n) = \mathcal{O}(g_1(n)g_2(n))$ .
3. Si  $p(n)$  est un polynôme de degré  $d$ , alors  $p(n) = \mathcal{O}(n^d)$ .

Il est également possible de démontrer les relations suivantes :

Cela nous permet d'évaluer la complexité d'expressions plus complexes telles que les suivantes

**Exemple 76.** Évaluons le comportement asymptotique de  $f(n) = 3n \log n + (n^2 + 3) \log n$ .

Par le théorème précédent, on sait que  $3n = \mathcal{O}(n)$ . Aussi,  $\log n = \mathcal{O}(\log n)$ , donc  $3n \log n = \mathcal{O}(n \log n)$ .

D'autre part,  $(n^2 + 3) = \mathcal{O}(n^2)$  et  $\log n = \mathcal{O}(\log n)$ , ce qui entraîne  $(n^2 + 3) \log n = \mathcal{O}(n^2 \log n)$ . Finalement, on a que  $f(n) = \mathcal{O}(\max(n \log n, n^2 \log n)) = \mathcal{O}(n^2 \log n)$ .

On conclut ce chapitre en mentionnant quelques complexités fréquemment rencontrés pour les algorithmes :

- $\mathcal{O}(1)$  (complexité constante) : trouver le maximum dans un tableau trié, échanger deux valeurs dans un tableau, incrémenter un compteur, etc.
- $\mathcal{O}(\log n)$  (complexité logarithmique) : fouille binaire, recherche dans un arbre équilibré, etc.

- $\mathcal{O}(n)$  (complexité linéaire) : fouille linéaire, recherche du maximum dans un tableau non trié, etc.
- $\mathcal{O}(n \log n)$  (complexité  $n \log n$ ) : tri par fusion, tri par tas, tri rapide (complexité moyenne), etc.
- $\mathcal{O}(n^2)$  (complexité quadratique) : tri par insertion, tri à bulle, etc.
- $\mathcal{O}(2^n)$  (complexité exponentielle) : décider si une expression logique est une tautologie, etc.
- $\mathcal{O}(n!)$  (complexité factorielle) : problème du voyageur de commerce (solution naïve).



# Chapitre 9

## Induction mathématique

Lorsqu'on conçoit des algorithmes et qu'on utilise des structures de données performantes, il est essentiel de développer une aptitude pour réfléchir de façon récursive. D'une part, de nombreuses structures de données discrètes sont construites naturellement de cette façon, et, d'autre part, plusieurs problèmes peuvent être résolus plus efficacement s'ils sont étudiés d'un point de vue récursif.

### 9.1 Principe de l'induction

Il est fréquent de devoir démontrer des énoncés de la forme  $P(n)$ , pour une infinité de nombres naturels  $n$ . Dans certains cas, une démonstration directe, par contradiction ou cas par cas est suffisante, mais parfois, il faut recourir à une technique de démonstration appelée *induction mathématique* ou *démonstration par récurrence*.

D'un point de vue logique, l'induction mathématique peut être simplement vue comme une application infinie du *modus ponens*. Pour simplifier, supposons que nous souhaitions démontrer la proposition  $\forall n P(n)$ , où  $P(n)$  est un prédicat quelconque dont l'univers de discours est l'ensemble  $\mathbb{N}$ . Supposons de plus que nous ayons démontré les deux propositions suivantes :

- (i)  $P(0)$  et
- (ii)  $\forall n P(n) \rightarrow P(n+1)$ .

Alors, par la partie (ii) et en prenant  $n = 0$ , nous avons que  $P(0) \rightarrow P(1)$ . En combinant (i) et cette dernière proposition, la règle du modus ponens nous permet de déduire que  $P(1)$  est vraie. Aussi, la partie (ii) implique que  $P(1) \rightarrow P(2)$  en prenant  $n = 1$ . On peut donc appliquer le modus ponens une seconde fois et déduire que  $P(2)$  est vraie.

Ce processus peut être répété indéfiniment, pour tout entier positif  $n$ . On en conclut donc que l'énoncé  $\forall n P(n)$  est vrai, puisque  $P(n)$  est vrai pour tout  $n \in \mathbb{N}$ . C'est ce qu'on appelle l'*induction mathématique*.

On utilise souvent une analogie basé sur les dominos pour expliquer le principe. Supposons qu'on ait une rangée infinie de dominos étiquetés par  $0, 1, 2, \dots, n, \dots$ . Supposons en outre que les deux propositions suivantes sont vérifiées :

- (i) Le premier domino tombe;
- (ii) Si le domino  $n$  tombe, alors le domino  $n+1$  tombe aussi.

Alors le principe d'induction nous permet de conclure que tous les dominos tomberont.

## 9.2 Démonstrations arithmétiques

Nous illustrons maintenant la technique de démonstration par induction sur deux exemples simples issus de l'arithmétique. Lorsqu'on démontre une proposition par induction, il est typique de diviser la présentation en trois parties (cas de base, hypothèse d'induction et étape inductive).

Nous commençons par démontrer formellement une formule vue au chapitre 4.

**Proposition 7.** Pour tout entier  $n \geq 0$ , on a

$$\sum_{i=0}^n i = \frac{n(n+1)}{2}.$$

**Démonstration.** Par induction sur  $n$ .

CAS DE BASE. On vérifie que l'énoncé est vrai pour  $n = 0$ . En effet, on obtient

$$\sum_{i=0}^n i = \sum_{i=0}^0 i = 0 = \frac{0(0+1)}{2} = \frac{n(n+1)}{2}.$$

HYPOTHÈSE D'INDUCTION. Supposons que pour tout entier  $n \geq 0$ , on ait bien

$$\sum_{i=0}^n i = \frac{n(n+1)}{2}.$$

ÉTAPE INDUCTIVE. On doit montrer que l'énoncé est toujours vrai pour  $n+1$ , c'est-à-dire que l'identité

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$

est vérifiée. Or,

$$\begin{aligned} \sum_{i=0}^{n+1} i &= \sum_{i=0}^n i + (n+1) \\ &= \frac{n(n+1)}{2} + (n+1) \quad (\text{par hypothèse d'induction}) \\ &= (n+1) \left[ \frac{n}{2} + 1 \right] \\ &= (n+1) \cdot \frac{n+2}{2} \\ &= \frac{(n+1)(n+2)}{2}, \end{aligned}$$

tel que voulu.

Le principe d'induction est aussi très utile pour démontrer des inégalités.

**Proposition 8.** Pour tout entier positif  $n \geq 0$ , on a  $n < 2^n$ .

**Démonstration.** Par induction sur  $n$ . Aussi, pour une raison expliquée un peu plus bas, il est nécessaire de démontrer deux cas de base ici plutôt qu'un.

CAS DE BASE. Si  $n = 0$ , on a bien

$$n = 0 < 1 = 2^0 = 2^n.$$

De la même façon, pour  $n = 1$ , on trouve

$$n = 1 < 2 = 2^1 = 2^n.$$

HYPOTHÈSE D'INDUCTION. Supposons que pour tout  $n \geq 1$ , on ait  $n < 2^n$ .

ÉTAPE INDUCTIVE. On doit montrer que  $n + 1 < 2^{n+1}$ . Or,

$$\begin{aligned} 2^{n+1} &= 2 \cdot 2^n \\ &> 2 \cdot n \quad (\text{par hypothèse d'induction}) \\ &= n + n \\ &\geq n + 1, \end{aligned}$$

ce qui entraîne que  $2^{n+1} > n + 1$ . À noter que les deux cas de base sont nécessaires, car on doit utiliser l'hypothèse  $n \geq 1$  pour obtenir la dernière inégalité.

## 9.3 Démonstrations combinatoires

Les démonstrations par inductions ne se limitent pas aux identités basées sur l'arithmétique. Nous avons vu au chapitre 2 une formule simple qui met en relation la cardinalité d'un ensemble fini  $A$  et la cardinalité de l'ensemble de tous ses sous-ensembles  $2^A$ . Nous démontrons maintenant cette identité.

**Proposition 9.** Soit  $A$  un ensemble fini de  $n$  éléments,  $n \geq 0$ . Alors  $|2^A| = 2^n$ , c'est-à-dire que  $A$  possède  $2^n$  sous-ensembles.

**Démonstration.** Par induction sur  $n$ .

CAS DE BASE. Si  $n = 0$ , alors  $A = \emptyset$ . On a bien que

$$|2^A| = |2^\emptyset| = |\{\emptyset\}| = 1 = 2^0 = 2^n.$$

HYPOTHÈSE D'INDUCTION. Supposons que pour tout ensemble  $A'$  de  $n$  éléments, on ait  $|2^{A'}| = 2^n$ .

ÉTAPE INDUCTIVE. On doit montrer que tout ensemble  $A$  de  $n + 1$  éléments vérifie  $|2^A| = 2^{n+1}$ . Soit  $A$  un ensemble de  $n + 1$  éléments,  $a \in A$  un élément quelconque de  $A$  et  $B = A - \{a\}$ . Nous souhaitons compter tous les sous-ensembles de  $A$ . On peut les diviser en deux catégories : (1) les sous-ensembles de  $A$  qui incluent  $a$  et (2) les sous-ensembles de  $A$  qui excluent  $a$ .

- (1) Tout sous-ensemble  $E$  de  $A$  qui inclut  $a$  est de la forme  $E' \cup \{a\}$ , où  $E'$  est un sous-ensemble de  $B$  quelconque. Or, il y a  $2^n$  possibilités de choix pour  $E'$ , par hypothèse d'induction, de sorte qu'il y a  $2^n$  sous-ensembles de  $A$  qui incluent  $a$ .
- (1) Tout sous-ensemble  $E$  de  $A$  qui exclut  $a$  est un sous-ensemble de  $B$ . On sait qu'il y en a  $2^n$  par hypothèse d'induction, ce qui entraîne qu'il y a  $2^n$  sous-ensembles de  $A$  qui excluent  $a$ .

Au total, il y a donc bel et bien  $2^n + 2^n = 2^{n+1}$  sous-ensembles de  $A$ .

## 9.4 Induction forte

Nous avons vu que dans certaines démonstrations par induction, il est nécessaire de démontrer plus d'un cas de base. Parfois, même démontrer plusieurs cas de base n'est pas suffisant : il faut également utiliser une hypothèse d'induction plus forte.

L'idée est la suivante. Supposons que nous souhaitions démontrer un énoncé de la forme  $\forall n P(n)$ , où  $n \in \mathbb{N}$ . On démontre d'abord certains cas de base (le nombre peut varier entre 1 et plusieurs). Ensuite, on suppose que  $P(k)$  est vraie pour tout  $k < n$ . Autrement dit, plutôt que de supposer vrai seulement  $P(n)$  pour démontrer  $P(n+1)$ , on suppose que tout est vrai jusqu'à  $n$  (exclusivement), et il ne nous reste alors qu'à conclure que  $P(n)$  aussi doit être vrai. C'est ce qu'on appelle l'*induction forte* ou le *second principe d'induction*.

Si on reprend l'analogie des dominos, l'*induction forte* peut-être interprétée de la façon suivante. Si

- (i) Le premier domino tombe (ou les  $d$  premiers dominos,  $d$  étant le nombre de cas de base),
- (i) Le fait que les  $k$  premiers dominos tombent, où  $k < n$ , implique que le domino  $n$  tombe également,

alors on peut conclure que tous les dominos tomberont. D'un point de vue logique, l'induction simple et l'induction forte sont équivalentes.

## 9.5 Application de l'induction forte

Nous illustrons maintenant l'induction forte sur deux exemples.

Rappelons que la suite de Fibonacci est définie récursivement par  $f_0 = f_1 = 1$  et  $f_n = f_{n-1} + f_{n-2}$  pour  $n \geq 2$ . En étudiant la parité des premiers termes

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$$

on constate que le motif (impair, impair, pair) semble se répéter indéfiniment. Nous allons le montrer formellement, c'est-à-dire que nous allons montrer que pour tout  $n \geq 0$ ,

$$f_n \text{ est } \begin{cases} \text{pair,} & \text{si } n \bmod 3 = 2; \\ \text{impair,} & \text{sinon.} \end{cases}$$

CAS DE BASE. On a bien que  $f_0$  et  $f_1$  sont tous deux impairs, ce qui montre que la proposition est vraie pour  $n = 0$  et  $n = 1$ .

HYPOTHÈSE D'INDUCTION. Supposons que pour tout  $m < n$ , on ait que

$$f_m \text{ est } \begin{cases} \text{pair,} & \text{si } m \bmod 3 = 2; \\ \text{impair,} & \text{sinon.} \end{cases}$$

ÉTAPE INDUCTIVE. On doit montrer que la proposition est vérifiée pour  $n$ . Or, par définition, nous savons que  $f_n = f_{n-1} + f_{n-2}$ . Il suffit donc d'appliquer l'hypothèse d'induction à  $f_{n-1}$  et  $f_{n-2}$  pour déduire la parité de  $f_n$ . On distingue trois cas.

- (1) Supposons d'abord que  $n \bmod 3 = 0$ . Alors  $(n-1) \bmod 3 = 2$  et  $(n-2) \bmod 3 = 1$ . Par hypothèse d'induction, on a donc que  $f_{n-1}$  est pair et  $f_{n-2}$  est impair, ce qui entraîne que  $f_n$  est impair (la somme d'un nombre pair et d'un nombre impair est impaire).

- (1) Supposons maintenant que  $n \bmod 3 = 1$ . Alors  $(n - 1) \bmod 3 = 0$  et  $(n - 2) \bmod 3 = 2$ , de sorte que  $f_{n-1}$  est impair et  $f_{n-2}$  est pair, par hypothèse d'induction. On en conclut que  $f_n$  est impair.
- (1) Finalement, si  $n \bmod 3 = 2$ , alors  $(n - 1) \bmod 3 = 1$  et  $(n - 2) \bmod 3 = 0$ . Ceci entraîne, toujours par hypothèse d'induction, que  $f_{n-1}$  et  $f_{n-2}$  sont impairs, et donc que  $f_n = f_{n-1} + f_{n-2}$  est pair.

Nous concluons ce chapitre avec un dernier exemple.

**Proposition 10.** Pour tout naturel  $n \geq 8$ , il existe des naturels  $a$  et  $b$  tels que  $n = 3a + 5b$ .

**Démonstration.** Par induction forte sur  $n$ .

CAS DE BASE. On a

$$\begin{aligned} 8 &= 3 \cdot 1 + 5 \cdot 1 \\ 9 &= 3 \cdot 2 + 5 \cdot 0 \\ 10 &= 3 \cdot 0 + 5 \cdot 2, \end{aligned}$$

ce qui montre que l'énoncé est vrai pour  $n = 8, 9, 10$ .

HYPOTHÈSE D'INDUCTION. Supposons que pour tout naturel  $8 \leq k < n$ , il existe des naturels  $a', b'$  tels que  $k = 3a' + 5b'$ .

ÉTAPE INDUCTIVE. Nous devons montrer qu'il existe des naturels  $a$  et  $b$  tels que  $n = 3a + 5b$ . Remarquons que  $n - 3 < n$ . Par conséquent, il existe des naturels  $a', b'$  tels que  $n - 3 = 3a' + 5b'$ , par hypothèse d'induction appliquée à  $n - 3$ . On en déduit donc que  $n = 3a' + 5b' + 3 = 3(a' + 1) + 5b'$ . Clairement  $a' + 1$  et  $b'$  sont des naturels : il suffit donc de prendre  $a = a' + 1$  et  $b = b'$ , c'est-à-dire qu'on a bien que  $n$  s'exprime sous la forme  $3a + 5b$ .



# Chapitre 10

## Récurtivité

La récurtivité joue un rôle fondamental en informatique. Aux chapitres précédents, nous avons abordé certaines structures récurtives. Nous approfondissons maintenant cette notion.

**Définition 40.** Une *structure récurtive* est une *structure* dont la définition fait référence à elle-même.

### 10.1 Exemples

Plus objets mathématiques et informatiques se définissent de façon récurtive.

**Suites.** La suite de Fibonacci est définie récurtivement par

$$f_0 = 1, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n, \text{ pour } n \geq 0.$$

**Mots.** Il est souvent utile de définir des mots ou des chaînes de caractères de façon récurtive. Par exemple, considérons la suite de mots  $\{t_n\}_{n \in \mathbb{N}}$  définie par

$$\begin{aligned} t_0 &= 0 \\ t_n &= t_{n-1} \overline{t_{n-1}}. \end{aligned}$$

Alors on obtient les mots suivants :

$$\begin{aligned} t_0 &= 0 \\ t_1 &= 0\overline{0} = 01 \\ t_2 &= 01\overline{01} = 0110 \\ t_3 &= 0110\overline{0110} = 01101001 \\ t_4 &= 01101001\overline{01101001} = 0110100110010110 \\ &\vdots \end{aligned}$$

Le mot infini  $t_\infty = 0110100110010110 \dots$  est appelé *mot de Thue-Morse* et apparaît naturellement dans des problèmes mathématiques variés.

**Ensembles.** Il existe de nombreuses façons de définir des ensembles. Par exemple, soit  $E$  l'ensemble défini par les deux conditions suivantes :

- (i)  $3 \in E$ ;

(i) Si  $x, y \in E$ , alors  $x + y \in E$ .

On constate que  $6 \in E$ , puisque si on prend  $x = y = 3 \in E$ , alors  $3 + 3 = 6 \in E$ . En prenant 3 et 6, on déduit alors que  $9 \in E$ . Plus généralement, on a que  $E = 3\mathbb{Z}$ .

**Fonctions/opérations.** De nombreuses fonctions ou opérations sont définies de façon récursive. Par exemple, on définit la fonction *factorielle* par

$$n! = \begin{cases} 1, & \text{si } n = 0; \\ n(n-1)!, & \text{si } n \geq 1. \end{cases}$$

pour tout nombre naturel  $n$ . Par conséquent,

$$4! = 4 \cdot 3! = 4 \cdot 3 \cdot 2! = 4 \cdot 3 \cdot 2 \cdot 1! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 24.$$

L'opération d'image-miroir d'un mot  $w$  sur un alphabet  $A$  est formellement définie de façon récursive

$$\tilde{w} = \begin{cases} \varepsilon, & \text{si } w = \varepsilon; \\ \tilde{w}'a, & \text{si } w = aw', \text{ où } a \in A \text{ et } w' \text{ est un suffixe de } w. \end{cases}$$

En effet, on constate que

$$\widetilde{001} = \tilde{0}10 = \tilde{1}00 = \tilde{\varepsilon}100 = \varepsilon \cdot 100 = 100.$$

**Structures de données.** De nombreuses structures de données (arbres, graphes, listes, piles, etc.) sont des objets récursifs. Ils seront abordés plus en détails dans d'autres cours. Nous nous concentrons sur l'une d'entre elles, les *arbres binaires*, définis de la façon suivante. Un *arbre binaire* est

- (i) Soit un *arbre vide*, noté  $\emptyset$ ;
- (i) Soit un *noeud* rattaché à deux arbres binaires, appelés *sous-arbre gauche* et *sous-arbre droit*.

À l'aide de cette définition, il est possible de définir la *hauteur*  $h(A)$  d'un arbre binaire  $A$  par

$$h(A) = \begin{cases} 0, & \text{si } A \text{ est un arbre vide;} \\ 1 + \max(h(G), h(D)), & \text{sinon, où } G \text{ et } D \text{ sont les sous-arbres de } A. \end{cases}$$

Par exemple,

$$\begin{aligned} h \left( \begin{array}{c} \text{ } \\ \diagup \quad \diagdown \\ \text{ } \end{array} \right) &= 1 + \max \left( h \left( \bigcirc \right), \quad h \left( \begin{array}{c} \text{ } \\ \diagup \quad \diagdown \\ \text{ } \end{array} \right) \right) \\ &= 1 + \max \left( 1 + \max(h(\emptyset), h(\emptyset)), \quad 1 + \max(h(\bigcirc), h(\emptyset)) \right) \\ &= 1 + \max \left( 1 + \max(0, 0), 1 + \max(1 + \max(h(\emptyset), h(\emptyset)), 0) \right) \\ &= 1 + \max \left( 1, 1 + \max(1 + \max(0, 0), 0) \right) \\ &= 1 + \max \left( 1, 1 + \max(1, 0) \right) \\ &= 1 + \max \left( 1, 2 \right) \\ &= 1 + 2 \\ &= 3. \end{aligned}$$

## 10.2 Récursivité et induction

Comme les objets définis récursivement peuvent paraître plus abstraits, il est souvent nécessaire de démontrer des propriétés utiles sur ceux-ci. Les démonstrations par induction deviennent alors incontournables. Nous illustrons ce fait avec quelques exemples.

**Exemple 77.** Montrons que pour toute paire de mots  $u, v$  sur un alphabet  $A$ , on a toujours

$$\widetilde{uv} = \widetilde{v}u.$$

On peut démontrer le résultat par induction sur la longueur du mot  $u$ .

CAS DE BASE. Si  $|u| = 0$ , alors  $u = \varepsilon$  et donc

$$\begin{aligned} \widetilde{uv} &= \widetilde{\varepsilon}v \\ &= \widetilde{v} \\ &= \widetilde{v} \cdot \varepsilon \\ &= \widetilde{v} \cdot \widetilde{\varepsilon} \\ &= \widetilde{v}u. \end{aligned}$$

HYPOTHÈSE D'INDUCTION. On suppose que pour tout mot  $u$  de longueur  $n \geq 0$ , on a  $\widetilde{uv} = \widetilde{v}u$ .

ÉTAPE INDUCTIVE. On doit montrer que si  $u$  est de longueur  $n + 1$ , alors  $\widetilde{uv} = \widetilde{v}u$ . Soit  $u$  un mot de longueur  $n + 1$ . Alors  $u = aw$ , pour une certaine lettre  $a$  et un certain mot  $w$ . On obtient donc

$$\begin{aligned} \widetilde{uv} &= \widetilde{awv} \\ &= \widetilde{wva} \quad (\text{par définition de l'image-miroir}) \\ &= \widetilde{v}wa \quad (\text{par hypothèse d'induction}) \\ &= \widetilde{v}aw \quad (\text{par définition de l'image-miroir}) \\ &= \widetilde{v}u \end{aligned}$$

On peut utiliser une idée semblable pour démontrer certaines propriétés sur la hauteur d'un arbre.

**Théorème 11.** Soit  $A$  un arbre binaire de  $n$  noeuds et de hauteur  $h$ . Alors

$$\log(n + 1) \leq h \leq n.$$

**Démonstration.** Par induction sur la hauteur de l'arbre  $h$ .

CAS DE BASE. Si  $h = 0$ , alors  $A$  est un arbre vide. Par conséquent,  $n = 0$  et on a bien  $\log(n + 1) = \log 1 = 0 \leq 0 \leq 0$ .

HYPOTHÈSE D'INDUCTION. On suppose que tout arbre binaire de recherche  $A'$  de  $n'$  noeuds et de hauteur  $h' < h$  vérifie  $\log(n' + 1) \leq h' \leq n'$ .

RÉCURRENCE. Soit  $A$  un arbre binaire de recherche de  $n$  noeuds et de hauteur  $h$  et  $G, D$  les sous-arbres gauche et droite de  $A$ . De plus, soient  $n_G, n_D, h_G, h_D$  leur taille et leur hauteur. Alors  $h_G, h_D < h$  et donc on peut appliquer l'hypothèse d'induction. On obtient donc

$$\log(n_G + 1) \leq h_G \leq n_G \quad \text{et} \quad \log(n_D + 1) \leq h_D \leq n_D.$$

En additionnant les inégalités de droite, on trouve

$$h_G + h_D \leq n_G + n_D.$$

D'une part, on a

$$h = 1 + \max(h_G, h_D) \leq 1 + h_G + h_D \leq 1 + n_G + n_D = n,$$

et donc  $h \leq n$ . D'autre part,

$$\begin{aligned} \log(n+1) &= \log(n_G + n_D + 2) \\ &\leq \log(2 \max(n_G, n_D) + 2) \\ &= \log(2(\max(n_G, n_D) + 1)) \\ &= 1 + \log(\max(n_G, n_D) + 1) \\ &= 1 + \max(\log(n_G + 1), \log(n_D + 1)) \\ &\leq 1 + \max(h_G, h_D) \\ &= h, \end{aligned}$$

c'est-à-dire que  $\log(n+1) \leq h$ , ce qui termine la démonstration par induction.

### 10.3 Algorithmes récursifs

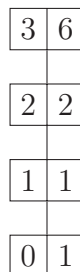
Nous étudions maintenant différents algorithmes récursifs, c'est-à-dire des algorithmes qui se font appel à eux-mêmes avec des paramètres plus petits. Il est important dans chaque cas de bien identifier les cas de base et, idéalement, d'en avoir un nombre minimum à des fins de lisibilité.

Tout d'abord, commençons par la fonction factorielle.

```

1: fonction FACTORIELLE( $n$  : naturel) : naturel
2:   si  $n = 0$  alors
3:     retourner 1
4:   sinon
5:     retourner  $n \cdot \text{FACTORIELLE}(n - 1)$ 
6:   fin si
7: fin fonction
```

On représente généralement la trace d'un algorithme récursif à l'aide d'un arbre, où on identifie les paramètres en entrée, les variables locales et les valeurs de retour. Par exemple, ici, on obtient



Nous avons vu il y a plusieurs cours un algorithme permettant de calculer le pgcd de deux nombres. Cet algorithme est naturellement récursif :

```

1: fonction PGCD( $a, b$  : naturels tels que  $a < b$ ) : naturel
2:   si  $a = 0$  alors
3:     retourner  $b$ 
4:   sinon
5:     retourner pgcd( $b \bmod a, a$ )
6:   fin si
7: fin fonction

```

À noter qu'il est important qu'on ait  $b \bmod a < a$  pour que l'algorithme fonctionne. C'est bien vrai, car le reste d'une division par  $a$  est toujours strictement inférieur à  $a$ .

Le même genre d'idée s'applique aux algorithmes sur des structures de données récursives. Par exemple, on calcule la hauteur d'un arbre binaire de la façon suivante :

```

1: fonction HAUTEUR( $A$  : arbre binaire) : naturel
2:   si  $A$  est un arbre vide alors
3:     retourner 0
4:   sinon
5:     retourner  $1 + \max(\text{HAUTEUR}(A.\text{GAUCHE}), \text{HAUTEUR}(A.\text{DROIT}))$ 
6:   fin si
7: fin fonction

```

On suppose qu'il est possible d'accéder aux sous-arbres gauche et droit en écrivant  $A.\text{GAUCHE}$  et  $A.\text{DROIT}$ .

Une application simple, mais très efficace, de la récursivité est l'*exponentiation rapide*. Supposons qu'on souhaite élever un nombre (ou une matrice)  $x$  à la puissance  $n$ . La stratégie naïve consiste à multiplier  $x$  avec lui-même  $n - 1$  fois. Or, si la multiplication est une opération coûteuse, il peut être intéressant d'en diminuer le nombre. On peut le faire de la façon suivante (sur les nombres réels, mais c'est aussi valide pour les matrices, par exemple) :

```

1: fonction EXPONENTIATION( $x$  : réel,  $n$  : naturel) : réel
2:   si  $n = 0$  alors
3:     retourner 1
4:   sinon si  $n$  est pair alors
5:      $y \leftarrow \text{EXPONENTIATION}(x, n/2)$ 
6:     retourner  $y \cdot y$ 
7:   sinon
8:      $y \leftarrow \text{EXPONENTIATION}(x, (n - 1)/2)$ 
9:     retourner  $x \cdot y \cdot y$ 
10:  fin si
11: fin fonction

```

Tous les algorithmes récursifs peuvent être réécrits de façon non récursive, mais le processus est parfois compliqué. Dans certains cas, l'efficacité est très différente. Pour le voir, écrivons le pseudocode d'un algorithme qui retourne le  $n$ -ième nombre de Fibonacci. L'algorithme récursif est très simple :

```

1: fonction FIBONACCI( $n$  : naturel) : naturel
2:   si  $n = 0$  ou  $n = 1$  alors
3:     retourner 1
4:   sinon
5:     retourner FIBONACCI( $n - 1$ ) + FIBONACCI( $n - 2$ )
6:   fin si

```

7: **fin fonction**

Une solution non récursive possible est la suivante :

```
1: fonction FIBONACCI( $n$  : naturel) : naturel
2:    $x \leftarrow 1$ 
3:    $y \leftarrow 1$ 
4:    $i \leftarrow 1$ 
5:   tant que  $i < n$  faire
6:      $z \leftarrow x + y$ 
7:      $x \leftarrow y$ 
8:      $y \leftarrow z$ 
9:      $i \leftarrow i + 1$ 
10:  fin tant que
11:  retourner  $y$ 
12: fin fonction
```

On constate que la version récursive effectue beaucoup de calculs redondants et donc inutiles. En fait, il est possible de montrer que la version récursive est  $\mathcal{O}(\phi^n)$ , où  $\phi \approx 1.61$  est le nombre d'or, alors que la version non récursive a une complexité  $\mathcal{O}(n)$ . En fait, il existe même une version basée sur l'algèbre matricielle qui permet de réduire la complexité à  $\mathcal{O}(\log n)$ .

# Chapitre 11

## Relations

Les relations et l'analyse des relations occupent une place fondamentale en mathématiques et en informatique. On les retrouve dans des applications très diversifiées :

- *Nombres.*  $=, \leq, \geq, <$  et  $>$ ;
- *Chaînes de caractères.*  $=, \leq, \geq$ , “être une sous-chaîne de”, “être un préfixe de”, etc.
- *Ensembles.*  $=, \subseteq, \subset, \supseteq, \supset$ , “être disjoint”, etc.
- *Arbres.* “être parent de”, “être enfant de”, “être frère/soeur de”, etc.
- *Modélisation.* On utilise des diagrammes UML pour représenter les relations “hérite de”, “utilise”, “fait partie de”, etc.
- *Web.* “page  $x$  pointe vers la page  $y$ ”
- *Réseaux sociaux.* “être adjacent à”, “être ami de”, “travailler avec”, “suivre sur Twitter”, “avoir accès aux publications Facebook de”, etc.

### 11.1 Représentation

La définition d'une relation est directement définie à l'aide de la théorie des ensembles.

**Définition 41.** Soient  $A$  et  $B$  deux ensembles. On dit que  $R$  est une *relation de  $A$  vers  $B$*  si  $R \subseteq A \times B$ . Souvent,  $A = B$  et on dit simplement que  $R$  est une *relation sur l'ensemble  $A$* .

À noter que  $\emptyset$  est une relation appelée la *relation vide*.

**Exemple 78.**  $R = \{(1, a), (2, b), (3, a)\}$  est une relation de  $\{1, 2, 3\}$  vers  $\{a, b\}$ .

La divisibilité  $|$  est une relation sur  $\mathbb{N}^*$ .

Soit  $\{0, 1\}^*$  l'ensemble des chaînes binaires. On écrit  $u \leq_{pref} v$  si  $u$  est un préfixe de  $v$ . Par exemple,  $01 \leq_{pref} 0100$ , mais  $01 \not\leq_{pref} 100$ . On a donc que  $\leq_{pref}$  est une relation sur  $\Sigma^*$ .

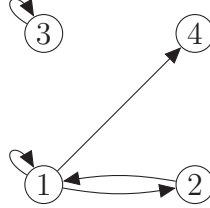
Soit  $P$  l'ensemble des pages sur le web. On peut écrire  $p_1 \rightarrow p_2$  si la page  $p_1$  fait référence à la page  $p_2$  (via `<href>`).

Lorsque le nombre d'éléments intervenant dans une relation n'est pas trop grand, il est pratique de représenter celle-ci par une matrice et/ou un graphe.

**Exemple 79.** Prenons la relation  $R = \{(1, 1), (1, 2), (1, 4), (2, 1), (3, 3)\}$  sur l'ensemble  $\{1, 2, 3\}$ . Alors

$$M_R = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Sous forme de graphe, on a plutôt

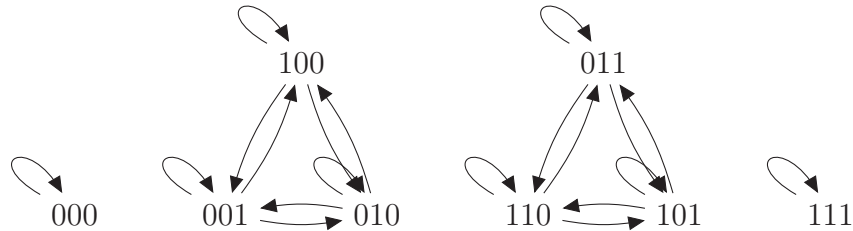


Cela fonctionne même si la relation n'est pas sur des nombres.

**Exemple 80.** Soit  $B_3$  l'ensemble des chaînes binaires de longueur 3 et  $R$  la relation définie par  $uRv$  si et seulement si  $u$  est un conjugué de  $v$ , c'est-à-dire que  $u$  et  $v$  sont les mêmes à rotation près si on les écrit sur un cercle. Alors

$$M_R = \begin{array}{l} \begin{matrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{matrix} \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Sous forme de graphe, on a



## 11.2 Propriétés

On voit que plusieurs relations ont des propriétés communes ou se ressemblent. Par exemple  $\leq$ ,  $\subseteq$  et même  $|$  ou encore  $=$  et “être une permutation cyclique de”. Nous nous intéressons maintenant à ces propriétés.

**Définition 42.** Soit  $R$  une relation sur un ensemble  $A$ . Alors on dit que  $R$  est

- *réflexive* si pour tout  $a \in A$ , on a que  $(a, a) \in R$ ;

- *irréflexive* si pour tout  $a \in A$ , on a que  $(a, a) \notin R$ ;
- *symétrique* si  $(a, b) \in R$  implique  $(b, a) \in R$ ;
- *antisymétrique* si  $(a, b), (b, a) \in R$  implique  $a = b$ ;
- *asymétrique* si  $(a, b) \in R$  implique  $(b, a) \notin R$ ;
- *transitive* si  $(a, b), (b, c) \in R$  implique  $(a, c) \in R$ .

Certaines relations particulières nous intéresseront :

**Définition 43.** Une relation  $R$  sur un ensemble  $A$  est

- une *relation d'équivalence* si elle est réflexive, symétrique et transitive;
- une *relation d'ordre* si elle est réflexive, antisymétrique et transitive.

**Exemple 81.** Considérons la relation  $|$  sur l'ensemble  $\mathbb{N}^*$ .

- Pour tout  $n \in \mathbb{N}^*$ , on a  $n | n$ . C'est donc une relation réflexive.
- Elle n'est pas irréflexive, puisqu'elle est réflexive et non vide.
- Elle n'est pas symétrique. Par exemple,  $2 | 4$ , mais  $4 \nmid 2$ .
- Elle est antisymétrique. Supposons que  $n | m$  et  $m | n$ . Alors il existe des naturels  $a$  et  $b$  tels que  $m = an$  et  $n = bm$ . On a donc  $m = an = a(bm) = (ab)m$ , ce qui entraîne  $ab = 1$  et alors  $a = b = 1$ , de sorte que  $m = n$ .
- Elle n'est pas asymétrique, puisqu'elle est réflexive et non vide.
- Elle est transitive. Supposons que  $n | m$  et  $m | p$ . Alors il existe des naturels  $a$  et  $b$  tels que  $m = an$  et  $p = bm$ . On a alors  $p = bm = b(an) = (ba)n$ , ce qui montre que  $n | p$ .
- $|$  est une relation d'ordre.

**Exemple 82.** Soit  $k > 0$  un entier. Considérons la relation  $R$  sur  $\mathbb{Z}$  définie par  $aRb$  si et seulement si  $a \equiv b \pmod{k}$ . Rappelons que, par définition,  $a \equiv b \pmod{k}$  si  $k | (a - b)$ .

- On a toujours  $a \equiv a \pmod{k}$  pour tout  $a \in \mathbb{Z}$ , de sorte que c'est une relation réflexive. En effet,  $k | 0 = a - a$ .
- Elle n'est pas irréflexive, puisqu'elle est réflexive et non vide.
- Si  $a \equiv b \pmod{k}$  alors  $b \equiv a \pmod{k}$ , ce qui entraîne que la relation est symétrique. En effet, si  $a \equiv b \pmod{k}$ , alors  $k | (a - b)$  et  $k | -(a - b) = b - a$ .
- Elle n'est pas antisymétrique, puisque  $0 \equiv k \pmod{k}$  et  $k \equiv 0 \pmod{k}$ .
- Elle n'est pas asymétrique, puisqu'elle est symétrique et non vide.
- Elle est transitive, puisque  $a \equiv b \pmod{k}$  et  $b \equiv c \pmod{k}$  implique  $a \equiv c \pmod{k}$ . En effet, si  $a \equiv b \pmod{k}$  et  $b \equiv c \pmod{k}$ , alors  $k | (a - b)$  et  $k | (b - c)$ . On a donc que  $k | [(a - b) + (b - c)] = a - c$ .
- C'est donc une relation d'équivalence.

Attention, certaines combinaisons de propriétés peuvent paraître surprenante. Par exemple, une relation peut être à la fois symétrique et antisymétrique. En effet, la relation  $R = \{(0, 0), (1, 1), (2, 2)\}$  sur  $\{0, 1, 2\}$  est symétrique et antisymétrique.

### 11.3 Relations d'équivalence

Une relation d'équivalence est une relation réflexive, symétrique et transitive. Les relations d'équivalence jouent un rôle très important en mathématiques et en informatique.

**Exemple 83.** Montrer que la relation “avoir le même parent que” sur l'ensemble des noeuds d'un arbre binaire différents de la racine est une relation d'équivalence.

*Réflexivité.* Un noeud a le même parent que lui-même. La relation est donc réflexive.

*Symétrie.* Si  $x$  a le même parent que  $y$ , alors  $y$  a le même parent que  $x$ .

*Transitivité.* Si  $x$  a le même parent que  $y$  et que  $y$  a le même parent que  $z$ , alors  $x$  a le même parent que  $z$ .

**Exemple 84.** Montrons que la relation “être conjugué” est une relation d'équivalence. Plus précisément, on écrit  $u \equiv v$  si et seulement s'il existe des mots  $x$  et  $y$  tels que  $u = xy$  et  $v = yx$ . Par exemple, 01001 et 00101 sont conjugués puisque  $u = 01 \cdot 001$  et  $v = 001 \cdot 01$  (on prend  $x = 01$  et  $y = 001$ ).

*Réflexivité.* Pour tout mot  $u$ , on a  $u \equiv u$ . En effet,  $u = u \cdot \varepsilon$  et  $u = \varepsilon \cdot u$ .

*Symétrie.* Supposons que  $u \equiv v$ . Alors il existe  $x, y$  tels que  $u = xy$  et  $v = yx$ . Il existe donc  $x', y'$  tels que  $v = x'y'$  et  $u = y'x'$  (il suffit de prendre  $x' = y$  et  $y' = x$ ). Ainsi,  $\equiv$  est une relation symétrique.

*Transitivité.* Supposons que  $u \equiv v$  et  $v \equiv w$ . Alors il existe  $x, y, x', y'$  tels que  $u = xy$ ,  $v = yx$ ,  $v = x'y'$  et  $w = y'x'$ . En particulier,  $yx = v = x'y'$ . On distingue trois cas.

(1) Si  $|y| = |x'|$ , alors  $y = x'$  et  $x = y'$ , ce qui entraîne  $w = y'x' = xy = u$ . Or,  $\equiv$  est réflexive, de sorte que  $u \equiv w$ .

(2) Supposons maintenant que  $|y| < |x'|$ . Alors il existe un mot  $z$  tel que  $x' = yz$ , de sorte que  $yx = v = x'y' = yzy'$ , ce qui montre que  $x = zy'$ . On a donc

$$u = xy = (zy')y = z(y'y) \quad \text{et} \quad w = y'x' = y'(yz) = (y'y)z.$$

On en conclut donc que  $u \equiv w$ .

(3) Il reste le cas  $|y| > |x'|$ . Il existe alors un mot  $z$  tel que  $y = x'z$ , ce qui entraîne  $x'y' = v = yx = x'zx$ . On en déduit que  $y' = zx$ . Par conséquent,

$$u = xy = x(x'z) = (xx')z \quad \text{et} \quad w = y'x' = (zx)x' = z(xx'),$$

ce qui montre que  $u \equiv w$ .

Un dernier exemple.

En géométrie, on dit que deux polygones ou figures sont *isométriques* s'il existe une rotation, une translation, une réflexion ou une réflexion glissée qui les lie.

Il est clair que c'est une relation réflexive, car une figure est reliée à elle-même via la translation de vecteur  $(0, 0)$ .

Il s'agit aussi d'une relation symétrique, car toute isométrie est inversible. Par exemple, si  $F$  est obtenu de  $F'$  par une translation de vecteur  $(a, b)$ , alors  $F'$  est obtenu de  $F$  par une translation de vecteur  $(-a, -b)$ . S'il s'agit plutôt d'une rotation d'angle  $\theta$ , alors on applique la rotation d'angle  $-\theta$ . Pour les réflexions, il suffit de les appliquer une seconde fois et pour les réflexions glissées, il suffit d'appliquer la même réflexion et la translation opposée.

Il s'agit aussi d'une relation transitive, mais c'est plus complexe à démontrer, car il faut vérifier que la combinaison d'une réflexion, rotation, translation, réflexion glissée avec une autre réflexion,

rotation, translation, réflexion glissée donne toujours une réflexion, rotation, translation, réflexion glissée. Nous ne le faisons pas ici, car c'est un exercice assez long (pas très difficile, mais long).

Soit  $R$  une relation d'équivalence sur un ensemble  $A$ . Alors  $R$  induit une partition naturelle de  $A$  que nous étudions maintenant.

Tout d'abord, pour tout  $a \in A$ , on écrit  $[a]_R$  (ou simplement  $[a]$ ) pour dénoter sa *classe d'équivalence*, définie par

$$[a]_R = \{b \in A \mid aRb\}$$

Autrement dit,  $[a]$  est l'ensemble des éléments équivalents à  $a$ . À noter que  $a \in [a]$  et donc une classe d'équivalence n'est jamais vide.

**Exemple 85.** Prenons la relation  $a \equiv b \pmod{5}$  sur l'ensemble  $\mathbb{Z}$ . Alors

$$[0] = \{\dots, -10, -5, 0, 5, 10, \dots\}$$

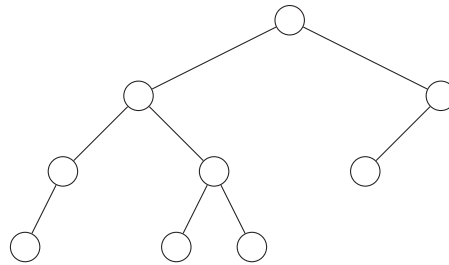
$$[1] = \{\dots, -9, -4, 1, 6, 11, \dots\}$$

$$[2] = \{\dots, -8, -3, 2, 7, 12, \dots\}$$

$$[3] = \{\dots, -7, -2, 3, 8, 13, \dots\}$$

$$[4] = \{\dots, -6, -1, 4, 9, 14, \dots\}$$

**Exemple 86.** Encerclons les classes d'équivalence dans l'arbre ci-bas :



**Exemple 87.** Pour la relation “être conjugué de”, on a

$$[ab] = \{ab, ba\}$$

$$[abaab] = \{abaab, baaba, aabab, ababa, babaa\}$$

$$[aaa] = \{aaa\}$$

**Exemple 88.** Si on prend la relation “être isométrique à” sur l'ensemble de polygones du plan cartésien et si  $C$  est le carré dont les sommets sont  $(0,0)$ ,  $(1,0)$ ,  $(0,1)$  et  $(1,1)$ , alors  $[C]$  est l'ensemble de tous les carrés dont le côté est de longueur 1. Il faut donc prendre tous les translatés et les rotatés de  $C$ , mais il n'est pas nécessaire de considérer les réflexions ni les réflexions glissées puisqu'elles n'ont aucun effet sur  $C$ .

Un théorème fondamental à propos des classes d'équivalence est le suivant :

**Théorème 12.** Soit  $R$  une relation sur un ensemble  $A$ . Alors

$$\{[a] \mid a \in A\}$$

forme une partition de l'ensemble  $A$ , c'est-à-dire que

- (i) Toute classe d'équivalence est non vide;
- (i) Pour tout  $a, b \in A$ , soit  $[a] = [b]$  ou  $[a] \cap [b] = \emptyset$ ;
- (i)  $\bigcup_{a \in A} [a] = A$ .

## 11.4 Combinaisons de relations

Comme les relations sont des ensembles, il est possible de les combiner à l'aide des opérateurs ensemblistes :  $\cup$ ,  $\cap$ ,  $-$ , etc.

Il existe aussi une opération de composition, très similaire à la composition de fonctions.

**Définition 44.** Soit  $R$  une relation de  $A$  vers  $B$  et  $S$  une relation de  $B$  vers  $C$ . La *composition* de  $S$  et de  $R$  est définie par

$$S \circ R = \{(a, b) \mid \text{il existe } c \in B \text{ tel que } (a, c) \in R \text{ et } (c, b) \in S\}$$

Si  $R$  est une relation sur un ensemble  $A$  et  $n$  est un entier positif ou nul, alors on dénote par  $R^n$  la relation  $R \circ R \circ R \circ \dots \circ R$  ( $n$  fois).

**Exemple 89.** Considérons les relations  $R : \{1, 2, 3\} \rightarrow \{a, b\}$  et  $S : \{a, b\} \rightarrow \{x, y, z\}$  définies par

$$\begin{aligned} R &= \{(1, b), (2, a), (2, b), (3, a)\} \\ S &= \{(a, x), (a, y), (a, z), (b, x), (b, z)\}. \end{aligned}$$

Alors

$$S \circ R = \{(1, x), (1, z), (2, x), (2, y), (2, z), (3, x), (3, y), (3, z)\}.$$

**Exemple 90.** Soit  $R = \{(1, 2), (2, 3), (3, 1), (3, 3)\}$  la relation sur  $\{1, 2, 3\}$ . Alors

$$\begin{aligned} R^1 &= \{(1, 2), (2, 3), (3, 1), (3, 3)\} \\ R^2 &= \{(1, 3), (2, 1), (2, 3), (3, 1), (3, 2), (3, 3)\} \\ R^3 &= \{(1, 1), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\} \\ R^4 &= \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 3)\} \\ R^5 &= \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\} \end{aligned}$$

**Exemple 91.** Soit la relation  $R$  définie sur  $\mathbb{Z}$  par

$$xRy \text{ si et seulement si } |x - y| = 1.$$

Alors  $R^2$  est la relation définie par

$$xR^2y \text{ si et seulement si } |x - y| = 2$$

et, plus généralement,  $R^n$  est la relation définie par

$$xR^ny \text{ si et seulement si } |x - y| = n.$$

## 11.5 Clôture/fermeture des relations

Parfois, certaines relations n'ont pas les propriétés souhaitées. Heureusement, il existe une opération simple permettant de transformer n'importe quelle relation pour qu'elle ait la propriété souhaitée.

**Définition 45.** Soit  $R$  une relation sur un ensemble  $A$ . La *clôture/fermeture* de  $R$  par rapport à une propriété  $p$  est la plus petite relation  $R'$  ayant la propriété  $p$  et contenant  $R$ .

**Exemple 92.** Prenons la relation  $R = \{(1, 1), (1, 2), (2, 1), (3, 2)\}$  sur  $\{1, 2, 3\}$ . La relation  $R$  n'est pas réflexive, car  $(2, 2) \notin R$ , mais

$$R^{refl} = R \cup \{(2, 2), (3, 3)\} = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 2), (3, 3)\}.$$

est bien réflexive et contient  $R$ .

**Exemple 93.** Considérons la relation  $<$  sur l'ensemble  $\mathbb{Z}$ . Cette relation n'est pas symétrique (elle est même asymétrique), puisque par exemple  $2 < 3$ , mais  $3 \not< 2$ . Alors

$$<^{sym} = \{(a, b) \mid a < b\} \cup \{(b, a) \mid a < b\} = \{(a, b) \mid a \neq b\}.$$

Autrement dit,  $<^{sym}$  est la même chose que  $\neq$ .

Plus généralement, on obtient les clôtures réflexive, symétrique et transitive de la façon suivante :

**Théorème 13.** Soit  $R$  une relation sur un ensemble  $A$ . Alors

1. la *clôture/fermeture réflexive* de  $R$  est la relation

$$R^{refl} = R \cup \{(a, a) \mid a \in A\}.$$

2. la *clôture/fermeture symétrique* de  $R$  est la relation

$$R^{sym} = R \cup \{(b, a) \mid (a, b) \in R\}.$$

3. la *clôture/fermeture transitive* de  $R$  est la relation

$$R^{trans} = R \cup R^2 \cup R^3 \cup \dots$$

La clôture transitive d'une relation peut être calculée à l'aide de l'algorithme de Warshall, que nous n'aborderons pas dans ce cours. Par contre, il est facile de visualiser les clôtures sur des graphes à l'aide de la méthode suivante :

1. La clôture *réflexive* est obtenue en ajoutant des boucles à tous les sommets qui n'en ont pas;
2. La clôture *symétrique* est obtenue en ajoutant des arcs de retour à tous les arcs qui n'en ont pas;
3. La clôture *transitive* est obtenue en ajoutant un arc directement entre deux sommets chaque fois qu'il existe un chemin entre ces deux sommets.