Structure Discrète
IFT1065
Algorithmes
Graphes et arbres
Relations
Fonctions
Cardinalité des ensembles

Franz Girardin

10 Décembre 2023

Table des matières

Récursion et algorithmes pt. 1

1.1 Type de données récursif §Lehman et al 7.1

Les *types de données récursifs* sont spécifiés par des **définitions récursives**; ces définitions indiquent comment construire des éléments de données à partir d'éléments de données précédents. À chaque type de données récursif est associé :

1. Des définitions de **propriétés** ou **fonctions**
2. Des méthode d'**induction structurelle** pour prouver que toutes les données du type indiqué ont une certain *propriété*.

Chaque définition récursive d'un type de données a deux partie : **Le(s) cas de base** qui spécifie que certains *éléments mathématiques connus* sont dans le type de données; **Le(s) constructeur(s)** qui spécifie comment construire de nouveaux *éléments de données* à partir des éléments de base.

Définition 1 Definition récursive d'une *chaîne de caractères*

Soit un **ensemble non nul** appelé un *alphabet*, dont ses éléments sont des *caractères* (ou *letttres*, *symboles*, *chiffres*), le **type de données récursif** A^* dépendant de l'alphabet A est définit comme suit :

- **Cas de base** : la chaîne de caractère vide, λ est dans A^*.
- **Cas constructeur** : Si $a \in A$ et $s \in A^*$, alors la paire $\langle a, s\rangle \in A^*$

Cette définition décrit toutes les *chaînes de caractères* possibles se trouvant dans A^* en se servant de l'alphabet A. La définition indique que si un élément se trouve dans l'alphabet et qu'une chaîne de caractère s se trouve dans l'ensemble A^* basé sur l'alphabet A, alors, il existe une autre chaîne de caractère dans A^* telle que la chaîne s est concaténée à l'élément a; donc, une autre chaîne s' commençant par a et finissant par s existe également dans A^*.

Ainsi, la chaîne de caractère 1011 peut être obtenue récursivement en commençant par la chaîne vide $\lambda \in A^*$:

$$\langle 1, \langle 0, \langle 1, \langle 1, \lambda\rangle\rangle\rangle\rangle$$

Définition 2 Définition récursive de la longueur d'une chaîne de caractère

La longueur $|s|$ d'une chaîne de caractère s, est définit récursivement comme suit :

- **Cas de base** : $|\lambda| ::= 0$
- Cas constructeur : $|\langle a, s\rangle| ::= 1 + |s|$

Définition 3 Concaténation

La *concaténation* $s \cdot t$ des chaînes de caractères $s, t \in A^*$ est définit récursivement comme suit :

1. **Cas de base** :
$$\lambda \cdot t ::= t$$
2. **Cas constructeur** :
$$\langle a, s\rangle \cdot t ::= \langle a, s \cdot t\rangle$$

Exemple 1

Considérons deux chaînes de caractères binaires $s = 01$ et $t = 11$. En utilisant la notation récursive, nous avons :

$$s = \langle 0, \langle 1, \lambda\rangle\rangle \quad \text{et} \quad t = \langle 1, \langle 1, \lambda\rangle\rangle$$

Pour concaténer s et t en utilisant la définition récursive, nous procédons comme suit :
1. Nous commençons avec la chaîne s et appliquons la règle du cas constructeur :

$$s \cdot t = \langle 0, \langle 1, \lambda\rangle\rangle \cdot \langle 1, \langle 1, \lambda\rangle\rangle$$

2. Nous appliquons ensuite la définition de la concaténation récursive :

$$\langle a, s'\rangle \cdot t = \langle a, s' \cdot t\rangle$$

3. En suivant cette règle, nous concaténons le reste de la chaîne s (qui est $\langle 1, \lambda\rangle$) avec t :

$$\langle 0, (\langle 1, \lambda\rangle \cdot t)\rangle$$

4. En utilisant le cas de base pour la concaténation ($\lambda \cdot t = t$), nous avons :

$$\langle 1, \lambda\rangle \cdot t = \langle 1, t\rangle$$

5. En remplaçant t par sa valeur, nous obtenons :

$$\langle 1, t\rangle = \langle 1, \langle 1, \langle 1, \lambda\rangle\rangle\rangle$$

6. En combinant tout, la représentation récursive de la chaîne concaténée $s \cdot t$ est :

$$\langle 0, \langle 1, \langle 1, \langle 1, \lambda\rangle\rangle\rangle\rangle$$

Ce qui correspond à la chaîne "0111" en notation standard.

1.2 Induction Structurelle §Lehman et al. 7.1.1

L'induction structurelle est une méthode qui permet de prouver que tous les éléments d'un type de données récursif ont une certaine *propriété*. Une preuve d'induction structurelle a **deux parties** :

— Prouver que chaque élément du cas de base *possède la propriété*

— Prouver que chaque élément du cas constructeur possède la propriété, lorsque le constructeur est appliqué sur les éléments qui possèdent la propriété.

Théorème 1 Principe d'induction structurelle

Soit P un prédicat sur un type de données définit récursivement R. **Si** :

— $P(b)$ est vrai pour chaque cas de base $b \in R$, et

— Pour tout constructeur à deux argument **c**,

$$[P(r) \text{ ET } P(s) \text{ IMPLIQUE } P(c(r,s))]$$

pour tout $r, s \in R$,
et pareillement pour tout constructeur prenant un autre nombre d'arguments,

alors,

$P(r)$ est vrai pour tout $r \in R$

Lemme 1 L'identité de droite de λ

$$\forall s \in A^*, s \cdot \lambda = s$$

Preuve 1

Soit l'affirmation

$$P(s) ::= [s \cdot \lambda = s]$$

Cas de base : $(s = \lambda)$ Nous montrons ici que la *propriété* $s \cdot \lambda = s$ est vraie **pour tous les éléments** appartenant s appartenant à A^*.

$$\begin{aligned} s \cdot \lambda &= \lambda \cdot \lambda \\ &= \lambda \\ &= s \end{aligned}$$

Cas constructeur : $(s = \langle a, t \rangle)$. Nous montrons ici que la *propriété* $s \cdot \lambda = s$ est vraie **pour tous les éléments** $s \in A^*$, lorsque le constructeur est apliqué sur les éléments $s \in A^*$.

$$\begin{aligned} s \cdot \lambda &= \langle a, t \rangle \lambda \\ &::= \langle a, t \cdot \lambda \rangle \\ &= \langle a, t \rangle \\ &= s \end{aligned}$$

1.3 Définir les naturels récursivement Lehman et al. §7.3

Définition 4 Entier naturel

L'ensemble des **entiers naturels** $\mathbb{N}$ est l'ensemble définit récursivement comme étant :

— $0 \in \mathbb{N}$

— **Si** $n \in \mathbb{N}$, **alors** le *successeur* $n+1$ de n est dans $\mathbb{N}$.

Définition 5 Factoriel

Le factoriel d'un nombre $n \in \mathbb{N}$ est définit récursivement comme étant :

— $fact(0) ::= 1$

— $fact(n+1) ::= (n+1) \ldots fact(n)$ pour $n \geq 0$.

1.4 Induction Mathématique §Hammack 14.1

L'induction mathématique vise à montrer qu'un prédicat $S(n)$ est vrai pour tout $n \in D \subseteq \mathbb{N}$. Il faut d'abord montrer qu'un **cas de base** S_1 est vrai. Dans **l'étape d'induction**, il faut montrer que le fait que S_k soit vrai force S_{k+1} à être vrai. Mathématiquement :

$$S_k \implies S_{k+1}$$

Grand lines d'une preuve par induction

Proposition :
Les propositions $S_1, S_2, S_3, S_4 \dots$ sont toutes vraies

Preuve. (Induction)

(1) Prouver que S_1 est vrai
(2) Prouver que pour tout entier $k \geq 1$, $S_k \implies S_{k+1}$ est vrai.

$\vdots$

Il s'ensuit que, *par induction mathématique*, S_n est vrai. □

1.5 Induction Mathématique forte §Hammack 14.3

L'induction mathématique forte vise à montrer qu'un prédicat $S(n)$ est vrai pour tout $n \in D \subseteq \mathbb{N}$. Il faut d'abord montrer qu'un ou plusieurs **cas de bases** $S_1, S_2, \dots S_r$ sont vrais. Dans l'étape d'induction, il faut assumer que les prédicats S_m sont vrais pour tous $1 \leq m \leq k$, pour **prouver que** $S(k+1)$ est vrai.

Grand lines d'une preuve par induction forte

Proposition :
Les propositions $S_1, S_2, S_3, S_n \dots$ sont toutes vraies

Preuve. (Induction)

(1) Prouver que $S_1, S_2, \dots S_r$ (les prédicats de base) sont vrais

(2.1) Assumer que les prédicats S_m sont vrais pour $1 \leq m \leq k$
(2.2) Prouver que pour tout entier $k \geq 1$, $S_k \implies S_{k+1}$ est vrai.

$\vdots$

Il s'ensuit que, *par induction mathématique* ***forte***, S_n est vrai. □

Théorème 2 Suite

Soit $a_1 = 1, a_2 = 3, a_k = a_{k-2} + 2a_{k-1}$. Il est vrai que a_k sera toujours impair.

Preuve 2

Soit S_n ::= *il est vrai que* a_n *est* ***impair***. Nous devons montrer que $\forall n \in \{3, 4, 5 \dots\}, S_n$.
Cas de base
Montrons d'abord que S_1 et S_2 sont vrais. Trivialement, $S_1 = 2$ et $S_2 = 3$ sont impairs.
Hérédité
Supposons que les prédicats S_m sont vrais où $2 \leq m \leq k$. Alors nous avons :

$$\begin{aligned} a_{k+1} &= a_{k-1} + 2a_k \\ &= a_{k-1} + 2(2c+1), c \in \mathbb{N} \\ &= a_{k-1} + 4c + 2 \\ &= a_{k-1} + 2(c+1) \end{aligned}$$

Nous savons que a_{k-1} est impair, *par l'hypothèse inductive que nous avons posé* et nous savons également que $2(c+1)$ est un nombre pair. Nous savons également que la somme de deux nombre de parité opposé engendre un nombre impair (Lemme). Par *induction mathématique forte*, il s'ensuit que a_{k+1} est impair. □

Théorème 3 Postage de timbre

N'imporete quel postage de 8 cents ou plus est possible en utilisant des timbes de 3 cents et des timbres de 5 cents.

Preuve 3

Soit S_n ::= *Il est vrai qu'un postage de n cents est possible en utilisant des timbres de 3 cents et des timbres de 5 cents.*
Nous devons montrer que $\forall n \in \{8, 9, 10, \dots, S_n\}$.
Case de base

Montrons d'abord que S_8, S_9 et S_{10} sont vrais. Trivialement,

S_8 : On peut utiliser 1 timbre de 5 cents et 1 timbre de 3 cents
S_9 : On peut utiliser 0 timbre de 5 cents et 3 timbre de 3 cents
S_{10} : On peut utiliser 2 timbre de 5 cents et 0 timbre de 3 cents

Hypothèse inductive

Soit $k \geq 10$, Supposons que S_m est vrai, pour tout m : $8 \leq m \leq k$. Par l'hypothèse inductive, S_{k-2} est vrai. Pour faire un postage de $k+1$ cents
nous pouvons utiliser un postage de $k-2$ cents et ajouter 1 timbre de 3 cents : on obtient un postage de $(k-2)+3 = k+1$ cents en utliisant des timbre de 3 et 5 cents. □

On voudrait montrer que la proposition :

$$\forall n \in \mathbb{N} fact - rec(n) = n!$$

Preuve 4

Comme on veut prouver la correction d'un algorithme récursif, on fait une preuve par induction.
Preuve.

Cas de base
Par définition, $fact - rec(0) = 1$. Et nous savons que $0! = 1$

Hérédité

$\forall n \in \mathbb{N}, fac - rec(n) = n! \implies fac - rec(n+1) = (n+1)!$

Preuve directe Soit $n \in \mathbb{N}$, supposong que $fac - rec(n) = n!$. Puisque $n + 1 \neq 0$,

$$\begin{aligned} fact - rec(n+1) &= (n+1) \times fact - rect(n+1-1) \\ &= (n+1) \times n! \\ &= (n+1)! \end{aligned}$$

1.6 Extraction d'algorithmes à partir de preuves

Toute entier naturel $n \geq 8$ peut s'écrire sous la forme de :

$$n = 5x + 3y \mid x, y \in \mathbb{N}$$

Autrement dit, $P(n)$:

$$\forall n \in 8, 9, \ldots, \exists x, y, \in \mathbb{N}, n = 5x + 3y$$

Preuve 5

Par induction mathématique forte.
Cas de base
Pour montrer $P(n+1)$ on utilise $P(n-2)$. Donc, si on montre seulement le cas de base $P(8)$, on ne peut pas déduire $P(9)$ ni $P(10)$. Il faut donc montrer deux autres cas de base.

Cas de base $P(8)$

$$\begin{aligned} P(8) &= 8 = 3 \times 1 + 5 \times 1 \\ P(9) &= 9 = 3 \times 3 + 5 \times 0 \\ P(10) &= 10 = 3 \times 0 + 5 \times 2 \end{aligned}$$

Hérédité

$\forall n \geq 8, (\forall k \in \mathbb{N}, 8 \leq k \leq n \implies P(n)) \implies P(n+1)$

Soit $n \geq 8$ et supposons P(1) pour tous les $k \in \{8, 9, \ldots\}$:

$$\begin{aligned} n + 1 &= (n-2) + 3 \\ &= 5x + 3y + 3 \\ &= 5x + 3(y+1) \end{aligned}$$

Pour faire une preuve d'existance non constructive, on peut procéder en faisant une preuve par l'absurde. Autrement dit :

$$\text{Supposons } \forall x \in D, \neg P(x)$$

Nous arrivons ensuite à une contradiction, ce qui montre que notre proposition originale,

$$\exists x \in D, P(x)$$

est **vraie**.

Note :
Toutes les preuves d'existence **contructives** cachent un algorithme.

Exemple 2

Entrée : $n \geq 8$
Sortie : $(x, y) \in \mathbb{N} \times \mathbb{N}$ tant que $n = 5x + 3y$ Decomp $5 - 3(n)$. À compléter. 1

Algorithmes

2.1 Variables et Algorithmes §Hammack 6.1

Définition 6 Algorithme

Un algorithme est *une séquence d'instructions* qui permettent d'engendrer un résultat désiré (**sortie**) à partir d'éléments de départ (**entrée**).

Définition 7 Variable

Une **variable** est un *symbol* auquel on peut assigner diverse **valeurs**. Lorsque nécessaire, on utilise les indices pour distinguer les variables : $\{x_1, x_2, \ldots, x_n\}$.

Dans les language de programmation, une variable est représente un *endroit* dans la mémoire de l'ordinateur qui peut posséder plusieurs valeurs, selon le moment. Par contre, à un temps donné, une variable **ne possède qu'une valeur**.

2.2 Assignation et Égalité §Hammack 6.1

Exemple 3

Il est possible de générer une commande telle que la valeur 7, par exemple, **est assignée** à la variable x. Dans ce cas, on utilise la syntaxe suivante.

$$x := 7$$

Concept. 1 Assignation et Égalité

Il faut savoir différencier une *assignation* d'une *égalité*. Si un algorithme effectue l'assignation $x := 7$ et que, plus tard dans l'algorithme apparaît la commande $x = 7$, le programme évaluera la véracité de la phrase $x = 7$. Cette phrase peut être soit **vraie** ou soit **fausse**, dépendamment de la valeur de x.

2.3 Boucles et Notation d'Algorithmes §Hammack 6.2

Les boucles sont des *séquences de commandes répétées* un certain nombre de fois.

Définition 8 Boucle `while`

Il s'agit d'une séquence de commandes s'exécutant **tant qu'** une certaine *condition* est respectée.

Algorithm 1: Simple boucle `while`

```
1 while P(x) do
2 |   Commande 1;
3 |   Commande 2;
  |   ⋮
4 |   Commande n ;
5 end
```

La boucle `while` exécute la séquence de $n-1$ commandes qu'elle contient pendant plusieurs *itérations*, jusqu'à ce que $P(x)$ soit **faux**.

Définition 9 Boucle `for`

Il s'agit d'une séquence de commande s'exécutant tant qu'une *condition définit par un intervalle précis* est respectée.

Algorithm 2: Simple boucle `while`

```
1 for i := m to n do
2 |   Commande 1;
3 |   Commande 2;
  |   ⋮
4 |   Commande n ;
5 end
```

2.4 Opérateurs Logiques dans les Algorithmes §Hammack 6.3

Il y a une relation étroite entre les algorithmes et la logique. Chaque boucle d'un algorithme est excutée à condition qu'une proposition $P(x)$ soit vraie. Or, cette proposition peut contenir plusieurs variables jointes par des opérateurs logiques.

Définition 10 Condition `if`

L'algorithme exécute les commande dans le corps de la condition `if` si celle-ci est respecté.

```
1 if P(x) then
2 |   Commande 1;
3 |   Commande 2;
  |   ⋮
4 |   Commande n ;
5 end
```

Définition 11 `if-else`

Une variante de du conditionnel `if` est le conditionnel `if-else`

```
1 if P(x) then
2 |   Commande
3 |   Commande
  |   ⋮
4 else
5 |   Commande
  |   ⋮
6 |
7 end
```

Exemple 4 Algorithme générant $n!$

Input : Un entier non nul n
Output: $n!$
1 **if** $n = 0$ **then**
2 | return 1 ..... parce que $0! = 1$
3 **else**
4 | $y := 1$
5 | **for** $i := 1$ ***to*** n **do**
6 | | $y := y \cdot i$
7 | **end**
8 | **output** y ..... parce que $y = n!$
9 **end**

Exemple 5 Trouve le plus grand membre d'une liste

Input : Une liste $X = (x_1, x_2, \ldots, x_n)$
Output: La plus grande entrée de la liste
1 $plusGrand := x_i$
2 **for** $i := 2$ ***to*** $i = n - 1$ **do**
3 | **if** $plusGrand < x_i$ **then**
4 | | $plusGrand := i$
5 | **end**
6 **end**
7 **output** `plusGrand`

Imaginons que nous voulons trier la liste de longeur 5 suivante $(5, 4, 3, 2, 1)$ en ordre croissant. En partant tu **premier terme** $(i = 1)$, nous **comparons** le premier terme au terme suivant $(i = 2)$. Si le terme est plus grand que le terme suivant, on les interchange. On obtient alors une nouvelle liste $(4, 5, 3, 2, 1)$. On continue avec le prochain terme $(i = 2)$ Si ce terme est plus grand que son successeur, on les interchange. On poursuit cette manipulation jusqu'au dernier terme et obtient alors la liste $(4, 3, 2, 1, 5)$. Ce processus est effectué pour un total de $n - 1$ fois. On peut généraliser cette approche pour une liste de longueur n

Algorithm 3: (Bubble sort)

Input : Une liste $X = (x_1, x_2, \ldots, x_n)$
Output: Cette même liste triée ordre cro/éissant
1 $j := longeurDeLaListe$
2 $i := longeurDeLaListe$
3 **for** $j := 1$ ***to*** $n - 1$ **do**
4 | **for** $i := 1$ ***to*** $j := n - 1$ **do**
5 | | **if** $x_{i+1} < x_i$ **then**
6 | | | `temporaire` $:= x_i$
7 | | | $x_i := x_{i+1}$
8 | | | $x_{i+1} :=$ `temporaire`
9 | | **end**
10 | **end**
11 **end**
12 **output** X

2.5 L'algorithme de division §Hammack 6.4

Théorème 4 Algorithme de Division

$$\forall a \in \mathbb{Z}, \exists b > 0, q, r : a = qb + r, 0 \leq r < b$$

L'algorithme de division indique que pour tout entier, il existe un entier positif b qui divise a tel que a est la somme qb addotionné au reste de la division, r. Pour implémenter cet algorithme, il suffit de constater que b divise a q fois. Autrement dit, soustraire b de a au plus, q fois. À la $(q + 1)$-*ième* fois, on obtient l'entier non nul r, soit le reste.

Algorithm 4: (Algorithme de Division)

Input : Des nombre $a \geq 0$ et $b > 0$
Output: Des entier q et r pour lesquels $a = qb + r$
1 $q := 0$
2 **while** $a > b$ **do**
3 | $a := a - b$
4 | $q := q + 1$
5 **end**
6 $r := a$
7 **output** q
8 **output** r

2.6 Procédures et Récursion §Hammack 6.5

Lorsqu'on écrit un algorithme, la répétition de code est à éviter. Pour éviter les répétions, on peut gérérer une **procédure** jouant le rôle de mini-algorithme ; cette procédure peut ensuite être appalée et réutilisée plusieurs fois. Ainsi, pour générer un algorithme qui calcule $C(n, k)$ on peut d'abord écrire une procédure pour obtenir $n!$. On applerait cette procédure `Fac(`n`)`.

Algorithm 5: Procédure `Fac(`n`)`

Input : Un nombre n
Output: $n!$
1 **if** $n = 0$ **then**
2 | retourne 1
3 **else**
4 | $y := 1$
5 | **for** $i := 1$ ***to*** $i := n$ **do**
6 | | $y := y \cdot i$
7 | **end**
8 | retourne y
9 **end**

Algorithm 6: Algorithme pour calculer $C(n,k)$

Input : Deux entier n et k avec ≥ 0
Output: $n!$

if $(k<0) \vee (k>n)$ **then**
  output 0
else
  output $\dfrac{\texttt{Fac}(n)}{\texttt{Fac}(k)\cdot\texttt{Fac}(n-k)}$
end

Définition 12 Procédure récursive

Nous savons qu'une procédure est un ensemble d'instruction accomplissant une tâche. Nous savons aussi qu'un algorithme peut appeler une ou plusieurs procédures. Une **procédure récursive** est un algorithme procédural qui s'appelle lui-même.

Algorithm 7: Procédure récurisve `RFac(`n`)`

Input : Un nombre n
Output: $n!$

if $n=0$ **then**
  retourne 1
else
  retourne $n\cdot\texttt{Rfac}(n-1)$
end

2.7 *Prouver qu'un algorithme est vrai* §Hammack 6.5

Lorsqu'on génère une *procédure récusrive*, il est possible il est possible de prouver qu'un algorithme est valide en utilisant l'induction.

Proposition 2.1 Validité de `RFac(`n`)`

Si n est un entier non nul, **alors** `RFac(`n`)` retourne la valeur exacte de, soit $n!$.

Preuve 6

Nous devons prouver lorsqu'on fournit un entier non nul n à la procédure `RFac(`n`)`, la procédure retourn la valeur $n!$. Nous allons procéder par *induction mathématique*.

Cas de base

Lorsque $n=0$, `RFac(`n`)` retourne 1 et nous savons que $0!=1$. Donc, l'algorithme retourne la valeur appropriée pour le cas de base.

Hypothèse inductive

Supposons que si n est un entier non nul, `RFac(`n`)` engendre la valeur $n!$.

Nous devons prouver que si l'hypothèse est vraie, `RFac(`$n+1$`)` engendre la valeur $(n+1)!$. Nous allons procéder par preuve directe. Selon la procédure `RFac(`n`)` :

$$\begin{aligned}\texttt{RFac}(n+1) &= (n+1)\texttt{RFac}((n+1)-1)\\ &= n\texttt{RFac}(n)+1\cdot\texttt{RFac}(n)\\ &= (n+1)\texttt{RFac}(n) \qquad\qquad = (n+1)n!\end{aligned}$$

L'hypothèse d'induction suggère que `RFac(`n`)` $=n!$, et donc $(n+1)$`RFac(`n`)` $=(n+1)n!$. L'algorithme retourne la valeur correcte de $(n+1)!$

Il s'ensuit, par induction que `RFac(`n`)` retorune la valeur valide de $n!$ pour tout $n\geq 0$.

Graphes et arbres

Un graphe est un système de **noeud**, *nodes* interconnectés. Les graphes peuvent être conceptualisés comme étant un ensemble de noeuds liés par *des lignes connectant noeuds*.

3.1 *Graphes et sous-graphes* §Hammack 15.1

Le **noeud** d'un graph est appelé sommet, *vertex* **en anglais**. Les graphes sont généralement dénotés par des lettre majuscules G ou H. Pour spécifier un graphe G, il faut décrire ses *sommets* et ses *arrêtes* (:= edges **en anglais**).

Définition 13 Un graphe

Un **graphe** G est un ensemble finit $V(G)$ d'objets appelés **sommets** ainsi que d'un ensemble $E(G)$ composés de sous-ensembles à deux éléments de $V(G)$ appelés **arrêtes**. N'importe quelle arrête $\{x,y\}\in E(G)$ est abbréviée xy ou yx.

Certains graphes sont si courant qu'on réserve des **symboles spéciaux pour eux**. C'est le cas des **graphes** P_n et C_n.

Définition 14

Un **chemin** P_n est le graphe ayant n sommets : $\{v_1, v_2, \ldots, v_n\}$. Le graphe possède les arrêtes arrêtes $E(P_n) = \{v_1v_2, v_2v_3, v_3v_4, \ldots, v_{n-1}v_n\}$.

Définition 15 Cycle

Un **cycle** C_n est le graphes ayant n sommet $\{v_1, v_2, \ldots, v_n\}$ et possédant les arrêtes $E(C_n) = \{v_1v_2, v_2v_3, v_3v_4, \ldots, v_{n-1}v_n, v_nv_1\}$

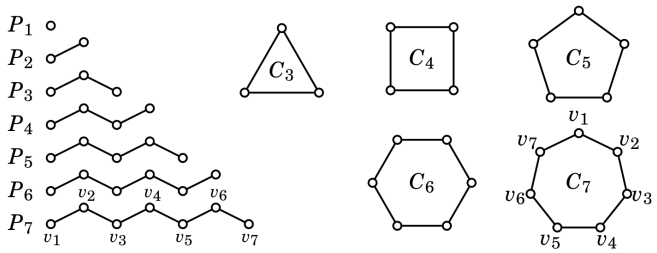


FIGURE 3.1 – Chemins P_n et cycles C_n

Un chemin ou un cycle est dit **pair** s'il possède un **nombre pair de sommets**. Autrement, il est dit **impair**.

Définition 16

Un **graphe complet** K_n est le graphe ayant n sommets et le nombre nécessaire d'arrêtes pour joindre toutes les paires de sommets. Le **nombre d'arrêtes** d'un graphe compelt K_n est donné par l'équation :

$$|E(K_n)| = \binom{n}{2} = \frac{n(n-1)}{2}$$

qui correspond au nombre de façons de choisir deux points à partir de n **sommets**

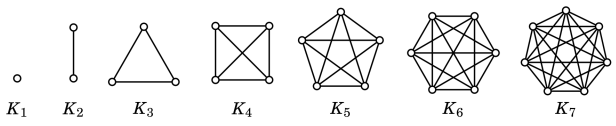


FIGURE 3.2 – Graphes complets

Définition 17 Graphes complet bipartis

Pour deux **entier positifs** m et n, le **graphe complet bipartis** $K_{m,n}$ est le graphe dont l'ensemble de sommets $V(K_{m,n}) = X \cup Y$ est l'union de deux **ensembles disjoints** X et Y de taille m et n, respectivement, et tels que $E(K_{m,n}) = \{xy : x \in X, y \in Y\}$.

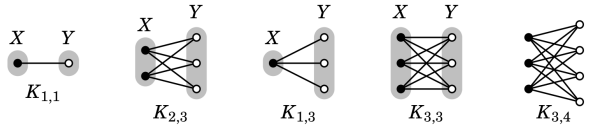


FIGURE 3.3 – Graphes Complets Bipartis

Dans un graphe complet bipartis, **chacun des sommet** $v \in X$ est connecté à un sommet à **chacun des sommets** $v \in Y$. Les graphes complets bipartis sont un cas spécial des **graphes bipartis**

Définition 18 Graphe bipartis

Un **graphe bipartis** est un graphe tel qu'il est possible de trouver une **partition** $V(G) = X \cup Y$ de $V(G)$ correspondant à deux ensembles disjoints, tel que chaque arrête G a un point dans X et un point dans Y.

Dans un graphe **bipartis incomplet**, chaque sommet $v \in X$ peut être connecté à un ou plusieurs sommets $v \in Y$. Par contre, il n'est pas vrai que tous les sommets $v \in X$ sont connecté à tous les sommets $v \in Y$.

Pour les graphes bipartis complet ou incomplet, un sommet $v \in X$ n'est jamais connecté à un autre sommet $v \in X$; les graphes qui on une arrête $E(G)$ connectant deux points appartenant à X ou deux points appartenant à Y ne sont pas bipartis.

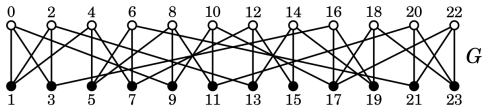


FIGURE 3.4 – Graphe bipartis incomplet

Par conséquent, les **cycles ayant un nombre impair de sommets** ne sont pas bipartis, puisqu'il n'est pas possible de diviser l'ensemble des sommet en deux ensembles disjoints ; il y aura toujours au moins un sommet $v_1 \in X$ connecté à un autre sommet $v_2 \in X$ ou un sommet $v_1 \in Y$ connecté à un autre sommet $v_2 \in Y$.

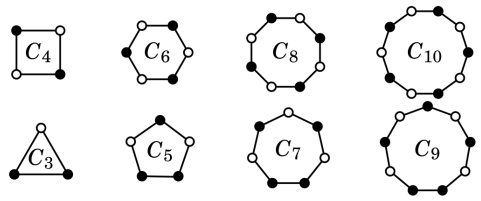


FIGURE 3.5 – Cycles bipartis pair et cycle non bipartis impairs

Définition 19 Sous-graphe

Un **graphe** H est dit être un **sous-graphe** de G si $V(H) \subset V(G)$ et $E(H) \subset E(G)$. Autrement dit, l'ensemble des **sommets** de H est composée d'un certain nombre de sommets de G et l'ensemble des arrête de H provient d'un certain nombre des arrête de G. On dit alors que G ***contient*** H.

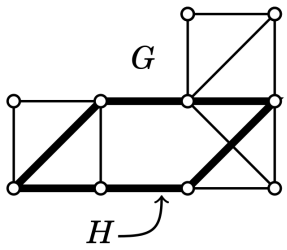


FIGURE 3.6 – H sous Graphe de G

Définition 20 Graphe connexe

Un **graphe** G est dit **connexe** si pour n'importe quel sommet x et y, il existe un chemin dans G qui commence à x et se termine à y. Un graphe qui n'est pas connexe est **non connexe**.

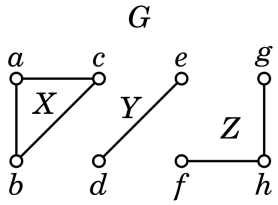


FIGURE 3.7 – Graphe G ayant trois composantes

Définition 21 Composante d'une graphe

Une **composante d'un graphe** est un graphe connexe qui n'est pas un sous-graphe d'un sous-graphe connexe plus large.

Par exemple, dans la figure 2.7, le graphe ab est un sous-graphes connexe, mais *n'est pas* une composante, puisque ab est un sous-graphe connexe d'un sous-graphe connexe plus large, X.

3.2 Sommets et degrés d'Arbres et Forêts

§HAMMACK 15.2

Définition 22 Degré d'un sommet

Soit un **sommet** x d'un **graphe** G, le degré de x, déonoté $\deg(x)$ est le nombre *d'arrêwes* de G incidantes à x. Pour le graphe suivant, on a :

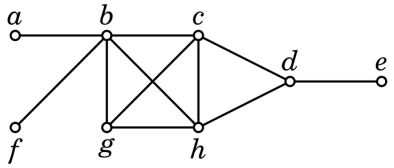


FIGURE 3.8 – Degré de sommets

$$\deg(a), \deg(b) = 5, \deg(c) = 4, \deg(d) = 3$$

La somme des degrés pour chacun des sommet est donnée par

$$\begin{aligned}\deg(a) + \deg(b) + \deg(c) + \deg(d) + \deg(e) \\ + \deg(f) \\ + \deg(g) \\ + \deg(h)\end{aligned}$$

Cela revient à faire la somme de tous les segments en gras dans le diagramme suivant. Et puisque chaque arrête possède **deux** segments gras, *la somme de tous les sommets d'un graph est égal à deux fois la sommes de ses arrêtes.*

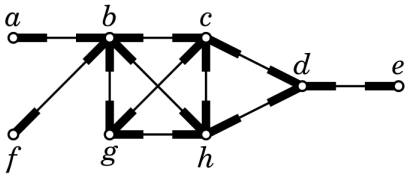


FIGURE 3.9 – Deux segments gras par arrêtes

Théorème 5 Degré d'un graphe

our chaque **graphe** G, la *somme des degrés* de ses sommets est **deux fois** la quantité d'arrête qu'il possède, c'est-à-dire,

$$\sum_{x \in V(G)} \deg(x) = 2|E(G)|.$$

Le corollaire de ce théorème est que la somme des degrés des sommets est *toujours paire*. Par ailleurs, nous savons que si la somme est paire, le nombre de degrés impair doit être paire.

Proposition 3.1

Un **graphe** a un nombre pair de sommets de degrés impair

Une identité parfois utile est la moeyennes des degrés d'un grpahes. Il est possible d'obtenir la *moyenne des degrés d'un graphe* en divisant la somme des degrés d'un graphe par la quantité de sommets :

$$\frac{\sum_{x \in V(G)} \deg(x)}{|V(G)|} = \frac{\deg(x) = 2|E(G)|}{V(G)}$$

Définition 23 Arbre et Forêt

Un **arbre** est un *graphe connexe* ne contenant aucun sous-graphe qui est un cycle. Une forêt est un graphe qui ne contient aucun cycle comme sous-graphe

Donc, une forêt est un graphe ayant des arbres comme composantes.

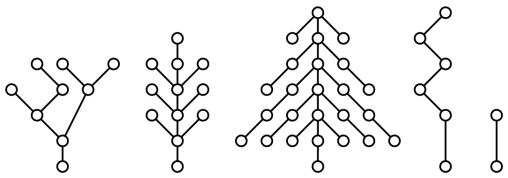


FIGURE 3.10 – Une forêt

Note :

Tous les **arbres** sont des **forêts** mais toutes les forêts *ne sont pas* des arbres.

Théorème 6

i un **arbre** a n sommets, alors il a $n-1$ **arrêtes**. Si une forêt a n **sommet**, alors elle a $n-c$ **arrêtes**, c étant le nombre de composantes de la forêts.

Lemme 2 Nombre de degrés d'un arbre

Un arbre ayant plus d'un sommet **possède** un sommet de degré 1. Par ailleurs, n'importe quel arbre ayant plus d'un sommet **possède** *deux* sommets de degré 1.

Note :

Voir Hammack **p.** 357 pour prouver le Théorème 3. Et voir Hammack **p.** 358 pour comprendre d'ou provient le Lemme 1.

3.3 Coloration d'un graphe et nombres chromatiques §Hammack 15.3

Concept. 2

Soit un entier k, une **k-coloration** d'un graphe est une assignation de k **couleur** aux sommets d'un graphe, *de sorte que chaque sommet obtienne une des k couleurs.*

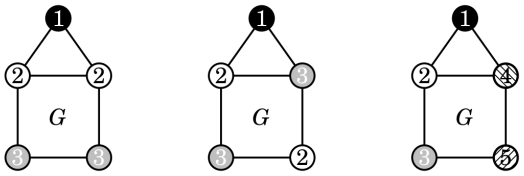


FIGURE 3.11 – Coloration de différents graphes. Seul le premier graphe possède une coloration propre.

Une **coloration** est dite *propre* si le graphe ne possède aucune sommets adjacent ayant la même couleur.

Définition 24 Nombre chromatique

Le **nombre chromatique** d'un graphe G, dénoté $\chi(G)$ et le plus petit entier k pour lequel le graphe G possède une k-coloration propre.

Note :

Chaque graphe complet K_n a un nombre chromatique de n :

$$\chi(K_n) = n$$

puisqu'on peut assigner une couleur différente à chacun des n sommets.

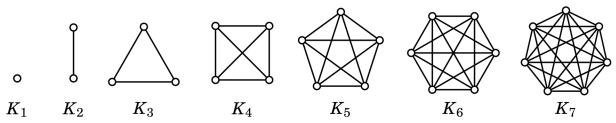


FIGURE 3.12 – Exemples de graphes complets

Note :

Chaque **graphe bipartis** $K_{m,n}$ a un nombre chromatique :

$$\chi(K_{mn}) = 2$$

puisqu'on peut imaginer chaque sommet $v \in X$ comme étant un coloré en noir et chque sommet $v \in Y$ comme étant coloré en blanc. Ainsi, il suffit de deux couleurs pour engendrer une coloration *propre*.

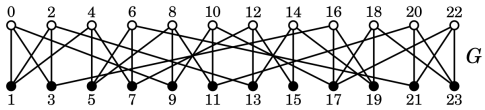


FIGURE 3.13 – Exemples de graphes bipartis

Note :

Chaque graphe cyclique C_n peut a un nombre chromatique de 2 ou 3, dépendamment de son nombre de sommet :

$$\chi(C_n) = \begin{cases} 2 & \text{si } n \text{ est pair} \\ 3 & \text{si } n \text{ est impair.} \end{cases}$$

Pour les graphes C_n impairs, on peut toujours alterner entre deux couleurs entre chaque sommet du cycle, jusqu'au dernier sommet ou on doit alors assigner une troisième couleur.

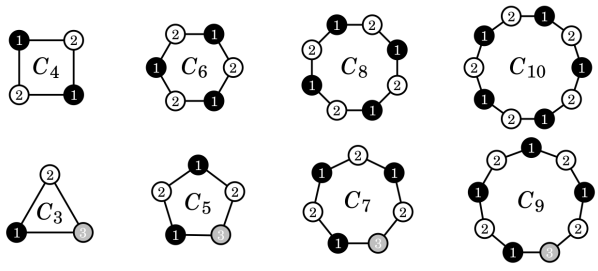


FIGURE 3.14 – Colorations de graphes cycliques C_n

Le consensus est qu'il n'existe pas de formule générale pour trouver le **nombre chromatique** d'un graphe au hasard, puisque ce problèeme est *NP-complet*. Il est cepndant possible de faire une *estimation* grâce à la proposition suivante qui permet de trouver la **borne inférireure** du nombre chromatique d'u graphe.

Proposition 3.2

Si H est un sous-graphe de G, alors $\chi(H) \leq \chi(G)$.

Le théorème de Brook permet de trouver la **borne supérieure** du nombre chromatique d'un graphe.

Théorème 7 Théorème de Brook

Supposons que G est un **graphe connexe** qui n'est ni un graphe complet, ni un graphe cyclique impair. Si le *degré du sommet le plus élevé de* G est Δ, **alors** $\chi(G) \leq \Delta$.

Marche Eulérienne

4.1 Marche Eulérienne §LEHMAN ET AL. 12.9.1

Concept. 3

Le concept de **marche Eulérienne** permet de répondre à des question telles que : *est-il possible de traverser chaque couloir du Musé des Beaux Arts exactement une fois ?*

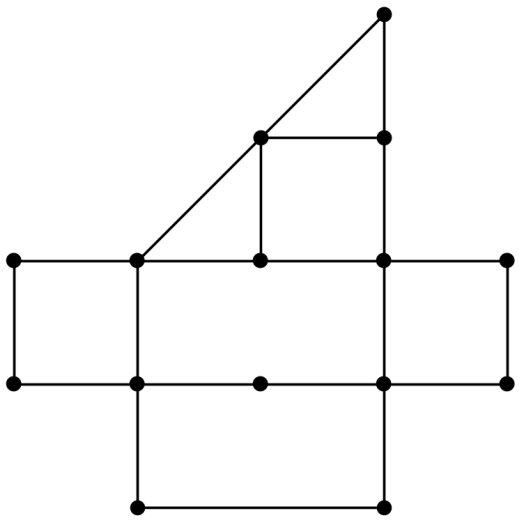


FIGURE 4.1 – Existe-t-il une marche qui inclus chaque arrête exactement *une fois*

Théorème 8 Marche Eulérienne

Un graphe connexe a une **marche Eulérienne** si et seulement si chaque sommet a un *degré pair*.

Théorème 9 Cycle d'Hamilton

Un *cycle Hamiltonien* dans un **graphe** G est un cycle qui visite chaque sommet de G exactement une fois. Similairement, un *chemin Hamiltonient* est un chemin dans un graphe G qui visite chaque sommet exactement une fois.

Relations et Fonctions

5.1 Relations §Hammack 16.1

Concept. 4 Relations

Les symboles tels que $<, \leq, =, |, \nmid, \geq, >, \in, \subset,$ etc. sont appelés *relations* puisqu'ils communiquent une *relation entre les entités*.

Définition 25 Relation

Une relation d'un ensemble A est un sous-ensemble $R \subseteq A \times A$. L'ensemble R contient donc des paires ordonnées (x, y) telles que $x, y \in A$ Nous faisons souvent l'abbréviation de $(x, y) \in R$ par xRy. L'affirmation $(x, y) \notin R$ est abbrévié $x \not R y$

Note :

$A \times A$ est espace de 2^e degré obtenu en effectuant le produit cartésien de A par lui-même ; il s'agit donc de *l'ensemble des paires ordonnées* de A par A. Une relation contient tous les les paires ordonnées et $A \times A$ ou seulement une certaine quantité d'entre elles : $R \subseteq A \times A$

Exemple 6 $\geq$

Soit $A = 1, 2, 3, 4$ et considéron l'ensemble suivant :

$$R = \Big\{(1,1),(2,1),(2,2),(3,3),(3,2),(3,1),(4,4),(4,3),(4,2),(4,1)\Big\} \subseteq A \times A$$

L'ensemble R est une relation sur A, par définition. Puisque $(1, 1) \in R$, nous avons $1R1$ Nous avons également $2R1$ et $2R2$, et ainsi de suite. Or, nous avons $3 \not R 4$. En realité, l'ensemble R décrit la relation $\geq$ pour l'ensemble A.

Exemple 7 Parité

Soit $A = 1, 2, 3, 4$ et considéron l'ensemble suivant :

$$S = \Big\{(1,1),(1,3),(3,1),(3,3),(2,2),(2,4),(4,2),(4,4)\Big\} \subseteq A \times A$$

En réalité, S exprime la relation *a la même parité que*. Nous savons que $(1, 3) \in S$ et donc $1S3$ se lit **1** *a la même parité que* **3**.

Exemple 8 Parité $\geq$

Constatons que :

$$R \cap S = \Big\{(1,1),(3,1),(3,3),(2,2),(4,2),(4,4)\Big\} \subseteq A \times A$$

En réalité, $x(R \cap S)y$ signifie $\boldsymbol{x \geq y}$, *et* $\boldsymbol{x}$ *et* $\boldsymbol{y}$ *ont la même parité.*

Note :

Les relations n'ont pas toujours de signification courante. Soit l'ensemble $B = 0, 1, 2, 3, 4, 5$, la relation

$$U = \Big\{(1,3),(3,3),(5,2),(2,5),(4,2)\Big\} \subseteq B \times B$$

ne contient *aucun sens dans le language courant*. Et pourtant, U est une relation de B

Il est possible de représenter une relation par un graphe ou une image. Par contre, toutes les relations ne peuvent pas *nécessairement* être représentées de façon visuelles. C'est le cas pour les relation qui impliquent des ensembles infinis.

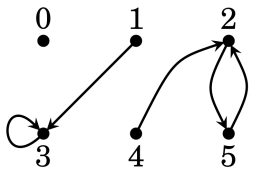


FIGURE 5.1 – Relation de $U \subseteq B \times B$

Exemple 9 Généralisation de $>$

Considérons l'ensemble :

$$R = \Big\{(x, y) \in Z \times Z : x - y \in \mathbb{N}\Big\} \subseteq Z \times Z$$

En réalité, R est relation ; R exprime la relation $>$.

5.2 Propriétés des relations §Hammack 16.2

Concept. 5 Relations en tant que propositions

Une relation xRy peut être interprétée comme une proposition ; elle est soit vraie ou soit fausse. Par exemple, $5 < 10$ est vraie et donc la relation $R \subseteq A$ qui définit le sens de $<$ peut être combinée à des opérateurs logiques pour former de nouvelles propositions :

$$xRy \implies yRx$$

La *proposition* ci-haut est formée à partir de la relation R et est donc soit vraie **ou** fausse.

Note :

Certaines relation ont des **propriétés** que d'autres relations n'ont pas Par exemple, la relation $\leq$ sur $\mathbb{Z}$ satisfait $x \leq x$, **alors que** la relation $<$ engendre un proposition xRx qui est fausse sur $\mathbb{Z}$, puisque il n'est jamais vrai que $x < x$.

Définition 26 *Propriétés* de relations

— R est dite *reflexive* si

$$\forall x \in A, xRx$$

— R est dite *symétrique* si

$$\forall x \in A, xRy \implies yRx$$

— R est dite *transitive* si

$$\forall x, y, z \in \mathbb{A}, \Big((xRy) \wedge (rRz)\Big) \implies (xRz)$$

Relation sur $\mathbb{Z}$	$<$	$\leq$	$=$	$\mid$	$\nmid$	$\neq$
Réflexif	non	oui	oui	oui	non	non
Symétrique	non	non	oui	non	non	oui
Transitif	oui	oui	oui	non	non	non

TABLE 5.1 – Relations sur $\mathbb{Z}$

Remarque 1 Déduction par diagramme

Les diagrammes de relations permettent d'indentifier rapidement les *propriétés* d'une relation.

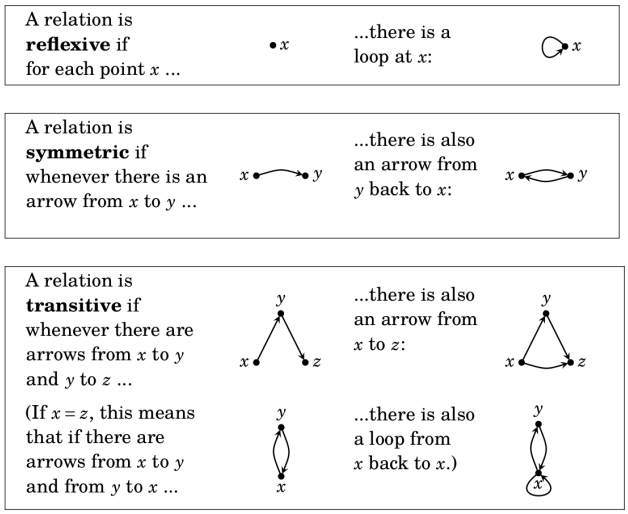


FIGURE 5.2 – Propriétés des relations (Hammack 16.2)

5.3 Relations d'équivalence §Hammack 16.3

Plusieurs relations telles que $=$ sont *réflexives* **et** *transitives* **et** *symétriques*. Une relation qui possède ces trois propriétés à la fois est appelé *relation d'équivalence*.

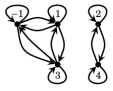
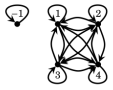
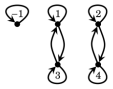
Relation R	Diagram	Equivalence classes (see next page)
"is equal to" (=) $R_1 = \{(-1,-1),(1,1),(2,2),(3,3),(4,4)\}$		$\{-1\}$, $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$
"has same parity as" $R_2 = \{(-1,-1),(1,1),(2,2),(3,3),(4,4),(-1,1),(1,-1),(-1,3),(3,-1),(1,3),(3,1),(2,4),(4,2)\}$		$\{-1,1,3\}$, $\{2,4\}$
"has same sign as" $R_3 = \{(-1,-1),(1,1),(2,2),(3,3),(4,4),(1,2),(2,1),(1,3),(3,1),(1,4),(4,1),(3,4),(4,3),(2,3),(3,2),(2,4),(4,2),(1,3),(3,1)\}$		$\{-1\}$, $\{1,2,3,4\}$
"has same parity and sign as" $R_4 = \{(-1,-1),(1,1),(2,2),(3,3),(4,4),(1,3),(3,1),(2,4),(4,2)\}$		$\{-1\}$, $\{1,3\}$, $\{2,4\}$

FIGURE 5.3 – Relations déquivalence selon Hammack 16.3

Lorsque R est une relation d'équivalence sur l'ensemble A, R divise A en sous-ensembles appelés des *classes équivalentes*.

Définition 27

Soit l'ensemble A, la relation R sur A et un certain élément $a \in A$, une **classe d'équivalence** est un sous-ensemble de A qui contient tous les éléments x tels que xRa est vrai. Ce sous ensemble est noté $[a]$; **Donc**, la classe d'équivalence contenant a est l'ensemble

$$[a] = \{x \in A : xRa\}$$

Exemple 10

Soit l'ensemble $A = \{1, 2, 3, 4\}$, la relation $=$ et l'entité $a \in A = 2$. La classe d'équivalence $[2]$ sur A est le sous-ensemble $\{x \in A : x = 2\}$. Puisque seul 2 a une telle relation avec lui-même, le seul élément de $[2]$ est 2 :

$$[2] = \{2\}$$

5.4 Classes d'équivalence et partitions §Hammack 16.4

$[a] = [b] \leftrightarrow aRb$ Suppossons que R est une relation d'équivalence sur un ensemble A. Et supposon que $a, b \in A$. **Alors**, $[a] = [b]$ *si et seulement si* aRb.

Preuve 7

Prouver que $[a] = [b] \implies aRb$

Supposons que $[a] = [b]$. Nous savons que aRa, *par la propriété réflexive* de R. **Donc**, a appartient à l'ensemble des nombres x tels que xRa. Mais cet ensemble, par définition, est [a]. Et par l'hypothèse d'origine, $[a] = [b]$. Ainsi, a appartient également à l'ensemble des nombres x tels que xRb (puisque $x \in [a] = [b]$). Ainsi, nous pouvons déduire que aRb. Nous résumons la logique ci-haut par l'expression suivante.

$$a \in \underbrace{\{x \in A : xRa\}}_{\text{def. de } [a]} = \underbrace{[a] = [b]}_{\text{hypothèse}} = \underbrace{\{x \in A : xRb\}}_{\text{def de } [b]}$$

Nous avons montré que, si $[a] = [b]$ (hypothèse), il s'ensuit que aRb.

Prouver que $aRb \implies [a] = [b]$

Supposons que aRb. Nous devons alors prouver que $[a] = [b]$. Nous procéderons en montrant que si $c \in [a]$, **alors** $x \in [b]$ **et que** si $c \in [b]$, **alors** $c \in [a]$.

1. Montrer que $\forall c \in \mathbb{Z}, c \in [a] \implies c \in [b]$

Supposons que $c \in [a]$. **Puisque** $c \in [a] = \{x \in A : xRa\}$, nous avons cRa. Par l'hypothèse d'origine, nous abons aRb et cRa (que nous venons de déduire), et, par conséquent bRa, puisque R est symétrique. Sachant que cRa et aRb, nous pouvons conclure que cRb, **puisque** R est transitif. Et nous savons que cRb signifie que $c \in \{x \in A : xRb\} = [b]$. **Et donc**, $x \in [b]$. Ainsi, nous venons de montrer que **si** $c \in [a]$, **alors** $c \in [b]$, pour tout $c \in \mathbb{Z}$. Cela signifie que $[a] \subseteq [b]$.

2. Montrer que $\forall c \in \mathbb{Z}, c \in [b] \implies c \in [a]$

Supposons que $c \in [b]$. Puisque $c \in [b] = \{x \in A : xRb\}$, nous avons cRb. **Or**, par l'hypothèse, nous avons aRb et cRb (que nous venons de déduire). Mais puisque R est réflexif, à partir de aRb, nous avons bRa. Maintenant nous avons cRb et bRa. Il s'ensuit alors que cRa, **puisque** R est transitif. Et nous savos que cRa signifie que $c \in \{x \in A : xRa\} = [a]$. Donc, nous concluons que $c \in [a]$. Ainsi, nous venons de montrer que si $c \in [b]$, **alors**, $c \in [a]$, pour tout $c \in \mathbb{Z}$. Cela signifie que $[b] \subseteq [a]$.

Nous venons de montrer que si $c \in [a]$, il s'ensuit que $[a] \subseteq [b]$ **et que** $[a] \subseteq [a]$. Cela signifie que $[a] = [b]$. □

5.5 Partitions §Hammack 16.4

Une relation d'étquivalence R sur un ensemble A divise A en plusieurs *classe d'équivalence*.

Définition 28 Partition

Une **partition** d'un ensemble A est un ensemble composé de sous-ensembles non vide de A tels que l'unions de ces sous-ensembles est égale à A, et l'intersection entre n'importe quelle paire de sous-ensemble faisant parti de cet ensemble est tojours $\emptyset$.

Autrement dit, aucun objet ne peut se trouver dans deux ou plusieurs sous-ensembles différents d'une partition de A.

Exemple 11 Partition

Soit $A = \{1, 2, 3, 4\}$. Une partition de A pourrait être :

$$\{\{1\}, \{2, 3\}, \{4\}\}$$

Théorème 10

Les classes d'équivalence d'un ensemble A forment une partition de cet ensemble.

Preuve 8

L'union de tous les ensembles $[a]$, des classes d'équivalence de A est donoté par

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

Nous devons donc prouver que $\bigcup_{a\in A}[a] = A$. Pour cela, il faut **1.** Montrer $\bigcup_{a\in A}[a] \subseteq A$ **et 2.** $A \subseteq \bigcup_{a\in A}[a]$

1. Montrer que $\bigcup_{a\in A}[a] \subseteq A$

Supposons que $x \in \bigcup_{a\in A}[a]$. **Alors**, nous savons que $x \in [a]$ pour un certain $a \in A$. Puisque $[a] \subseteq A$, il s'ensuit que $x \in A$. Notons que, par définition, tous les $x \in [a]$ doivent se trouver dans A, puisque $[a] = \{x \in A : xRa\}$. Nous venons de montrer que si $x \in \bigcup_{a\in A}[a]$, alors $x \in A$. Cela montre que $\bigcup_{a\in A}[a] \subseteq A$.

2. Montrer que $A \subseteq B\bigcup_{a\in A}[a]$

Supposons que $x \in A$. **Puisque** $x \in [x]$, nous savons **alors** que $x \in [a]$ pour une certain $a \in A$ (c'est-à-dire un $a = x$). **Et puisque** $x \in [a]$, il s'ensuit que $x \in \bigcup_{a\in A}[a]$.

Finalement, nous devons montrer que si $[a] \neq [b]$ **alors** $[a] \cap [b] = \emptyset$ Nous procédons par contraposé. Supposons que $[a] \cap [b] \neq \emptyset$, Ainsi, il existe une élément c tel que $c \in [a]$ et $c \in [b]$. **Nous savons que** $c \in [a]$ signifie que cRa et que aRc, puisque R est symétrique. **Pareillement**, $c \in [b]$ singifie que cRb. Nous avons alors aRc et cRb. **Et ainsi**, nous avons, aRb, **puisque** R **est transitif**. Par le théorème précédent, nous avons $aRb \implies [a] = [b]$. Nous venons de montrer la contraposée : $([a] \cap [b] \neq \emptyset) \implies [a] = [b]$.

Nous avons montré que $\bigcup_{a\in A}[a] = A$ et que deux sous-ensembles $[a] \neq [b]$ de A n'ont pas d'intersection. Il s'ensuit que le Théorème est vrai. □.

5.6 Entiers modulo n HAMMACK 16.5

$\forall n \in \mathbb{N}, \equiv (\mod n)$ est une relation réflexive, transitive et symétrique.

Les **classes d'équivalence** de la relation $\equiv (\mod 5)$ sont particulièrement importantes, puisqu'elles engendrent un système utile.

$$\begin{aligned}
[0] = \{x \in \mathbb{Z} : x \equiv 0(\mod 5)\} &= \{x \in \mathbb{Z} : 5|(x-0)\} \\
&= \{\dots, -10, -5, 0, 10, 15, \dots\} \\
[1] = \{x \in \mathbb{Z} : x \equiv 0(\mod 4)\} &= \{x \in \mathbb{Z} : 5|(x-1)\} \\
&= \{\dots, -9, -4, 1, 6, 11, 16, \dots\} \\
[2] = \{x \in \mathbb{Z} : x \equiv 2(\mod 5)\} &= \{x \in \mathbb{Z} : 5|(x-2)\} \\
&= \{\dots, -8, -3, 2, 7, 12, 17, \dots\} \\
[3] = \{x \in \mathbb{Z} : x \equiv 0(\mod 3)\} &= \{x \in \mathbb{Z} : 5|(x-3)\} \\
&= \{\dots, -7, -2, 3, 8, 13, 18\dots\} \\
[4] = \{x \in \mathbb{Z} : x \equiv 0(\mod 4)\} &= \{x \in \mathbb{Z} : 5|(x-4)\} \\
&= \{\dots, -6, -1, 4, 9, 14, 19, \dots\}
\end{aligned}$$

Note :

Ces classes d'équivalence forment une partition de l'ensemble $\mathbb{Z}$. **Par ailleurs**, $[0] = [5] = [10] = [15]$ *et ainsi de suite*. Nous savons aussi que $[1] = [6] = [11] = [16]\dots$

Concept. 6 Opérations sur $\mathbb{Z}_5$

Les cinq classes d'équivalences forment un ensemble appelé $\mathbb{Z}_5$ forment un système qui permet les opérations suivantes :

$$[a] + [b] = [a + b]$$
$$[a] \cdot [b] = [a \cdot b]$$

Définition 29 Entiers modudlo n

Soit $n \in \mathbb{N}$. Les classes d'équivalence de la relation d'équivalence $\equiv (\mod n)$ sont $[0], [1], [2], \dots, [n-1]$. Les **entiers modulo n** est l'ensemble $\mathbb{Z} = \{[0], [1], [2], \dots, [n-1]\}$. Les éléments de Z_n peuvent être additionnés avec la règle $[a] + [b] = [a + b]$ et $[a] \cdot [b] = [a \cdot b]$.

5.7 Relations entre les ensembles §HAMMACK 16.6

Définition 30 Relation entre deux ensembles

Une **relation** de l'ensemble A à l'ensemble B est le sous-ensemble $R \subseteq A \times B$. On peut, dans ce cas, abbrévier l'expression $(x, y) \in R$ par xRy, sachant que x et y appartiennent aux ensembles A et B, respectivement.

Fonctions

6.1 Fonction §Hammack 16.1

Exemple 12 x^2

Considérons la **fonction** suivante : $f(x) = x^2$. Nous pouvons alors réécrire f comme suit

$$R = \{(x, x^2) : x \in \mathbb{R}\} \subseteq \mathbb{R} \times \mathbb{R}$$

Exemple 13 $|n| + 2$

Considérons la fonction suivante : $f(x) = |n| + 2$. Nous pouvons alors réécrire f comme suite

$$R = \{(x, |x| + 2) : x \in \mathbb{Z}\} \subseteq \mathbb{Z} \times \mathbb{N}$$

Définition 31 Fonction

Supposons que A et B sont des ensembles. Une **fonction** f de A à B dénoté $f : A \to B$, est une relation $f \subseteq A \times B$ de A à B, satisfaisant la **propriété** que pour chaque $a \in A$, la relation f contient exactement une *paire ordonnée* (a, b). L'affirmation $(a, b) \in f$ est abbréviée $f(a) = b$.

Définition 32 Domaine, codomaine, étendue

Une fonction $f : A \to B$, l'ensemble A est appelé **domaine** de f. L'ensemble B est appelé le **codomaine** de f. L'ensemble des valeurs $b \in B$ qui sont telles que $f(a) = b$ est l'**image** de f.

Note :

Lorsque A et B contiennent un ensemble **fini** et suffisamment petit de valeurs, il est parfois utile de les représenter graphiquement.

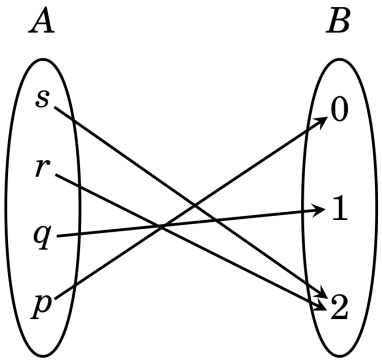


FIGURE 6.1 – Représentation graphique de la fonction $f = \{(p, 0), (q, 1), (r, 2), (s, 2)\}$

Définition 33 Égalité de deux fonctions

Deux fonctons $f : A \to B$ et $g : A \to D$ sont **égales** si $f(x) = g(x)$ pour tout $x \in A$.

6.2 Fonctions injectives et Surjectives §Hammack 16.2

Définition 34 Propriétés

Une fonction $f : A \to B$ est :

- **Injective** ou dite *1 pour 1* **si** pour tout $x, y \in A, x \neq y \implies f(x) \neq f(y)$;
- **Surjective** ou dite *mappé sur* **si** pour tout $y \in B$, il existe un x est tel que $f(x) = y$;
- **Bijective si** f est à la fois injective **et** surjective

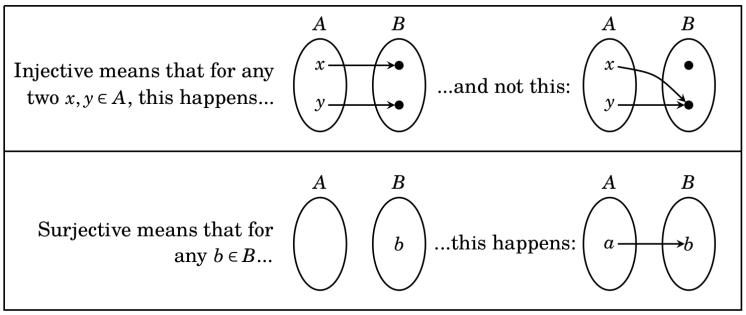


FIGURE 6.2 – Injectivité et surjectivité

Note :

Parfois il est nécessaire de **prouver qu'une fonction est injective**. On peut utiliser les deux approche qui suivent

— **Preuve directe** pour montrer que $f : A \to B$ est *injective*

▶ L'**objectif** est de montrer que :

$$\forall x, y \in A, (x \neq y) \implies f(x) \neq f(y)$$

▶ Supposer que $x, y \in A$ **et** $x \neq y$.

▷ Conclure que $f(x) \neq f(y)$.

— **Preuve par contraposée** pour montrer que $f : A \to B$ est *injective*

▶ Supposer que $x, y \in A$ **et** $f(x) = f(y)$.

▷ Conclure que $x = y$.

— **Preuve par contre-exemple** pour montrer que $f : A \to B$ **n'est pas** injective.

▶ Il faut réfuter la proposition :
$(x \neq y) \implies \Big(f(x) \neq f(y)\Big)$

▷ Il suffit de trouver deux **éléments** $x, y \in A$ pour lesquels $x \neq y$ **et** $f(x) = f(y)$.

— **Prouver** que $f : A \to B$ est *surjective*

▶ L'**objectif** est de montrer que :

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

▶ Supposer que $b \in B$.

▷ Montrer qu'il existe un $a \in A$ tel que $f(a) = b$.

— **Prouver** que $f : A \to B$ *n'est pas surjective*

▷ Montrer qu'il existe un b :

$$\exists b \in B, \forall a \in A, f(x) \neq f(b)$$

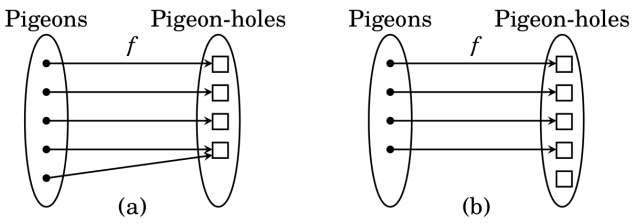
6.3 Reformulation du principe du pigeonnier

§Hammack 17.3

Concept. 7 Adaptation du principe

Supposons qu'un **ensemble** A de *pigeons* volent vers un ensemble B de *pigeonnier*. Ce phénomène peut être décris pas la fonction $f : A \to B$ où un pigeon X vole à l'intérieur d'un pigeonnier $f(X)$.

Nous constatons alors que si **le nombre de pigeon est supérieur** au nombre de pigeonnier, cela force **deux pigeons** à voler dans la même case ; $f : A \to B$ n'est donc pas *injective*. Nous constatons aussi que si **le nombre de piegons est inférireure** au nombre de piegonnier, il reste au moins un case vacante ; $f : A \to B$ n'est donc pas *surjective*.



Théorème 11 Le principe du pigeonnier

Supposons que A et B sont des ensemble fini et que $f : A \to B$ est une fonction quelconque. **Alors**,

— Si $|A| > |B|$, f *n'est pas* **injective**.

— Si $|A| < |B|$, f *n'est pas* **surjective**.

6.4 Composition §Hammack 17.4

Définition 35 Composition

Supposons que $f : A \to B, g : B \to C$ sont des **fonctions** ayant la propriété que le codomaine de f est le domaine de g. La **composition** de f avec g est une autre fonction, dénotée $g \circ f$ et suivant la règle suivante : Si $x \in A$, **alors** $g \circ f(x) = g(f(x))$. Par conséquent, $g \circ f$ envoit les éléments de A aux éléments de C ; $g \circ f : A \to C$.

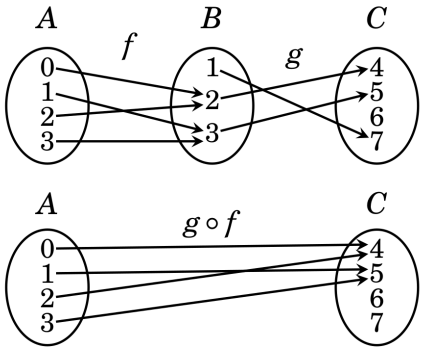


FIGURE 6.3 – Composition $g \circ f$

Note :

Pour que l'expression $g \circ f$ ait un sens, il faut que le codomaine de f soit égal au domaine de g ou qu'il soit au moins un sous-ensemble du domaine de g.

Théorème 12 Associativité de la composition

es composition de fonction sont **associatives**. C'est-à-dire que **si** $f : A \to B$, $g : B \to C, h : C \to D$, **alors** $(h \circ g) \circ f = h \circ (g \circ f)$.

Preuve 9

Supposons que f, g, h sont tels que mentionnées. Il s'ensuit que, *par la définition d'une* composition, $(h \circ g) \circ f$ **et** $h \circ (g \circ f)$ sont toute deux des **fonctions de** A à D. Pour montrer qu'elles sont égales, nous devons montrer que :

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

$$\Big((h \circ g) \circ f\Big)(x) = (h \circ g)(f(x)) = \underbrace{h(g(f(x)))}_{\textit{def. composition}}$$

$$\Big(h \circ (g \circ f)\Big)(x) = h \circ (g(f(x)) = \underbrace{h(g(f(x)))}_{\textit{def. composition}}$$

Donc, l'égalité tient, puisque les deux côté de l'égalité que nous devions prouver sont égal à $h(g(f(x)))$. □

Théorème 13

Supposons que $f : A \to B, g : B \to C$, **Si** f et g **sont toutes deux** *injectives*, **alors** $g \circ f$ est **injective**. **Si** f et g **sont toutes deux** *surjectives*, **alors** $g \circ f$ est **surjective**.

Preuve 10

Supposons que f et g sont **toutes deux** *injectives*. Pour montrer que la **composition** $g \circ f$ est injective, nous devons montrer que **si** $g \circ f(x) = g \circ f(y)$, **alors** $x = y$, nécessairement. Pour cela, supposons que $g \circ f(x) = g \circ f(y)$. Nous avons alors $g(f(x)) = g(f(y))$, *par la définition de compoosition*. **Ainsi**, nous avons l'égalité $f(x) = f(y)$. Pour que cette égalité soit vraie, il faut que $x = y$. Et puique f est injective ($f(x) = f(y)$ seulement lorsque $x = y$), nous savons que $x = y$, **nécessairement**; il n'existe pas de $y \neq x$ tel que $f(x) = f(y)$ à cause de la proprité d'injectivité de f. Par conséquent, $g \circ f$ est injective.

Supposons maintenant que g et f sont **toutes deux** *surjectives*. Pour montrer que la **composition** $g \circ f$ est surjective, nous devons montrer que n'importe pour n'impoorte quel élément $c \in C$, il existe un **élément correspondant** $a \in A$ tel que $g \circ f(a) = c$. Considérons une valeur arbibraire $c \in C$. Parce que g est surjectif, il y a un élément $b \in B$ pour lequel $g(b) = c$. Et puisque f est surjectif, il y a un élément $a \in A$ pour lequel $f(a) = b$. Par conséquent, $g(f(a)) = g(b) = c$, ce qui veut dire que $g \circ f(a) = c$. **Ainsi**, $g \circ f$ est **surjectif**. □

6.5 Fonctions inverses §Hammack 17.5

Définition 36 Fonction d'indentité

Soit une ensemble A, la **fonction d'identité** sur A est la fonction $i_A : A \to A$ définit par $i_A(x) = x$ pour tout $x \in A$.

Exemple 14

Si $A = \{1, 2, 3\}$, **alors** $i_A = \{(1,1), (2,2), (3,3)\}$. Par ailleurs, $i_{\mathbb{Z}} = \Big\{(n, n) : n \in \mathbb{Z}\Big\}$

Note :

La fonction d'identité d'une ensemble A, i_A est toujours bijective. Elle est injective puisque $i_A(x) = i_A(y)$ implique que $x = y$. Elle est surjective parce que si on prend n'importe quel élément b dans le codomaine de A, **alors** b est également dans le domaine de A, et $i_A(b) = b$.

Définition 37 Relation inverse

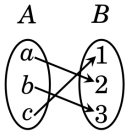
Soit une relation R de A à B, la **relation inverse de** R est la relation de B à A définit par $R^{-1} = \Big\{(y, x) : (x, y) \in R\Big\}$. **Aisi**, l'inverse de R est la relation R^{-1} obtenue en interchangeant les éléments dans chaque paire ordonnée de R.

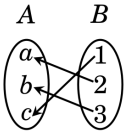
Exemple 15 Relation inverse

Soit $A = \{a, b, c\}$ et $b = \{1, 2, 3\}$, et supposons que f est la **relation inverse** $f = \Big\{(a, 2), (b, 3), (c, 1)\Big\}$ de A à B. **Alors**, $f^{-1} = \Big\{(2, a), (3, b), (1, c)\Big\}$ est la relation de B à A qui est la relation inverse de f.

Note :

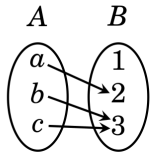
f est en fait une fonction de A à B et f^{-1} est une fonction de B à A.

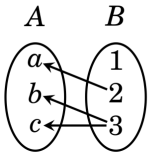




Exemple 16 Relation inverse

Considérons $g = \{(a,2),(b,3),(c,3)\}$ de A à B. **Alors,** $g^{-1} = \{(2,a),(3,b),(3,c)\}$. Bien que g est une fonction, g^{-1} n'est pas une fonction, *parce que g n'est pas* ***bijectif***





Théorème 14

Soit une **fonction** $f : A \to B$. **Alors,** f est **bijective** *si et seulement si* la relation inverse f^{-1} est une fonction de B à A.

Théorème 15

Si $f : A \to B$ est **bijective**, **alors** son *inverse* est la fonction $f^{-1} : B \to A$. Les fonctions f et f^{-1} obéissent aux l'équations $f^{-1} \circ f = i_A$ et $f \circ f^{-1} = i_B$.

Note :

Voir (faire) Exemples 17.12 et suite de 17.5 sur la même page

`§Hammack chapitre 17.5`

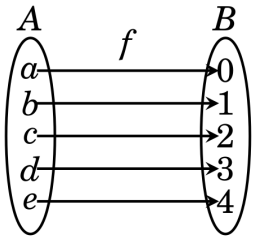
Cardinalité des ensembles

Note :

Deux ensembles $|A|$ et $|B|$ peuvent être **infini** tout en étant tels que $|A| < |B|$

Définition 38 Cardinalité d'un ensemble

Deux **ensembles** A et B ont la même cardinalité, $|A| = |B|$, s'il existe une fonction **bijective** telle que $f : A \to B$. Si une telle fonction n'existe pas, **alors,** les ensembles sont de **cardinalité inégale**, c'est-à-dire $|A| \neq |B|$.



Définition 39 Bijection

Une **bijection** est une *fonction bijective* qui permet de connecter A à B de façon bijective, c'est-à-dire $f : A \to B$ est une bijection des ensembles A et B. Similairement, une fonction *injective* et *surjective* est appelée **injection** et **surjection** de A et B, respectivement.

Théorème 16 Comparaison de $\mathbb{N}$ *et* $\mathbb{Z}$

Il existe une **bijection** $f : \mathbb{N} \to \mathbb{Z}$. Par conséquent $|N| = |Z|$.

n	1	2	3	4	5	6	7	8	9	...
$f(n)$	0	1	-1	2	-2	3	-3	4	-4	...

Preuve 11

Le tableau ci-haut montre que pour chaque élément de la bijection $f : \mathbb{N} \to \mathbb{Z}$ permet de relier les ensembles $\mathbb{N}$ et $\mathbb{Z}$ de façon bijective. Chaque **entier** apparaît une seule fois sur le 2^e rangée et soit un entier $b \in \mathbb{Z}$ il existe un entier $a \in \mathbb{N}$ tel que $f(a) = b$. Nous savons aussi que tous les éléments de $x \in \mathbb{N}$ ont une image unique dans $\mathbb{Z}$ en appliquant, f ; c'est-à-dire $(f(x) = f(y)) \implies x = y$. **Par définition,** f est une bijection de $\mathbb{N}$ à $\mathbb{Z}$ et donc, nous pouvons conclure que $|\mathbb{N}| = |\mathbb{Z}|$. □

Théorème 17

Il n'existe aucune **bijection** $f : \mathbb{N} \to \mathbb{R}$. Par conséquent $|N| \neq |R|$.

7.1 Esembles comptables HAMMACK 18.2

Définition 40 Ensemble comptable

Supposons que A est un **ensemble**. **Alors,** A est **dénombrablement infini** si $|\mathbb{N}| = |A|$, c'est-à-dire s'il existe une *bijection* $\mathbb{N} \to A$. L'ensemble A est **non dénombrable** si A est infini **et** $|\mathbb{N}| \neq |A|$, c'est-à-dire A est infini et il n'existe *aucune* bijection $\mathbb{N} \to A$.

Note :

$\mathbb{N}$ est dénombrablement infini mais $\mathbb{R}$ est non comptable.

Définition 41

La cardinalité de l'ensemble des entiers naturels est dénotée $\aleph_0$ C'est-à-dire $|\mathbb{N}| = \aleph_0$. **Ainsi,** tout les entiers

dénombrablement infini ont une cardinalité $\aleph_0$.

Exemple 17

Soit $E = \{2k : k \in \mathbb{Z}\}$, l'ensemble des entiers pairs. Nous savons que $f : \mathbb{Z} \to E$ définit par $f(n) = 2n$ est une **bijection** de $\mathbb{Z}$ à E. Par conséquent, $|\mathbb{Z}| = |E|$. Ainsi, $|\mathbb{N}| = |\mathbb{Z}| = |E|$. Donc l'ensemble E est dénombrablement infini.

Théorème 18

Un ensemble A est dénombrablement infini *si et seulement si* ses éléments peuvent être arrangé en une liste infinie $a_1, a_2, a_3, \ldots$.

Exemple 18

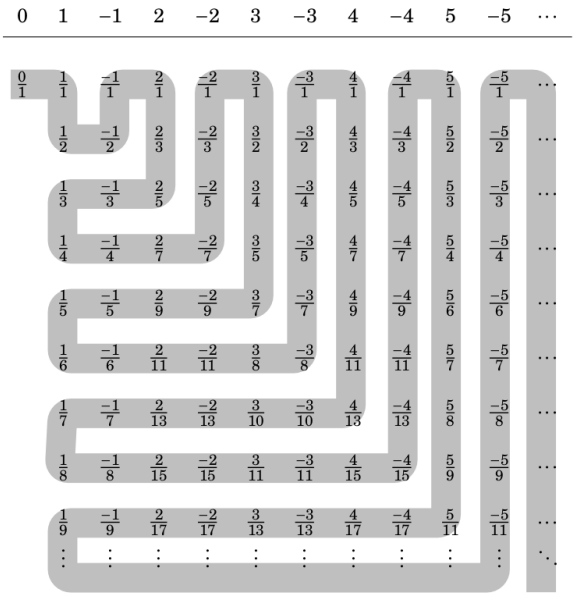
Soit P l'ensemble des nombre premiers. Puisque nous pouvons lister ses éléments par $2, 3, 4, 5, 7, 11, 13, \ldots$, il s'ensuit que P est dénombrablement infini. □

L'ensemble des nombre rationel, $\mathbb{Q}$ est dénombrablement infini.

Preuve 12

Il suffit de lister ces éléments de la façon suivante. On considère un tableau où on liste d'abord tous les entiers dans $\mathbb{Q}$ en commençant à 0 et en alternant entre des valeurs positives et négatives par la suite. Les rangées suivantes contiennent les fractions irréductible de $\mathbb{Q}$. Pour chaque colonne, un entier k forme le numérateur des fractions irrédutibles. La colonne 2, par exemple, contient les fractions $\frac{2}{1}, \frac{2}{3}, \frac{2}{5}, \frac{2}{7}$, etc. Elle ne contient pas $\frac{2}{2}, \frac{2}{4}, \frac{2}{6}$, etc. puisque celles-ci sont réductibles et, d'ailleurs, cette quantité est compté par la première rangée contenant $k = 2, k = 4, k = 6$, etc. Nous pouvons ensuite former un chemin commançant à $\frac{0}{1}$ et serpentant à travers les colonnes et rangées. Ce chemin représente la liste des éléments de $\mathbb{Q}$ □

0	1	−1	2	−2	3	−3	4	−4	5	−5	⋯
$\frac{0}{1}$	$\frac{1}{1}$	$\frac{-1}{1}$	$\frac{2}{1}$	$\frac{-2}{1}$	$\frac{3}{1}$	$\frac{-3}{1}$	$\frac{4}{1}$	$\frac{-4}{1}$	$\frac{5}{1}$	$\frac{-5}{1}$	⋯
	$\frac{1}{2}$	$\frac{-1}{2}$	$\frac{2}{3}$	$\frac{-2}{3}$	$\frac{3}{2}$	$\frac{-3}{2}$	$\frac{4}{3}$	$\frac{-4}{3}$	$\frac{5}{2}$	$\frac{-5}{2}$	⋯
	$\frac{1}{3}$	$\frac{-1}{3}$	$\frac{2}{5}$	$\frac{-2}{5}$	$\frac{3}{4}$	$\frac{-3}{4}$	$\frac{4}{5}$	$\frac{-4}{5}$	$\frac{5}{3}$	$\frac{-5}{3}$	⋯
	$\frac{1}{4}$	$\frac{-1}{4}$	$\frac{2}{7}$	$\frac{-2}{7}$	$\frac{3}{5}$	$\frac{-3}{5}$	$\frac{4}{7}$	$\frac{-4}{7}$	$\frac{5}{4}$	$\frac{-5}{4}$	⋯
	$\frac{1}{5}$	$\frac{-1}{5}$	$\frac{2}{9}$	$\frac{-2}{9}$	$\frac{3}{7}$	$\frac{-3}{7}$	$\frac{4}{9}$	$\frac{-4}{9}$	$\frac{5}{6}$	$\frac{-5}{6}$	⋯
	$\frac{1}{6}$	$\frac{-1}{6}$	$\frac{2}{11}$	$\frac{-2}{11}$	$\frac{3}{8}$	$\frac{-3}{8}$	$\frac{4}{11}$	$\frac{-4}{11}$	$\frac{5}{7}$	$\frac{-5}{7}$	⋯
	$\frac{1}{7}$	$\frac{-1}{7}$	$\frac{2}{13}$	$\frac{-2}{13}$	$\frac{3}{10}$	$\frac{-3}{10}$	$\frac{4}{13}$	$\frac{-4}{13}$	$\frac{5}{8}$	$\frac{-5}{8}$	⋯
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

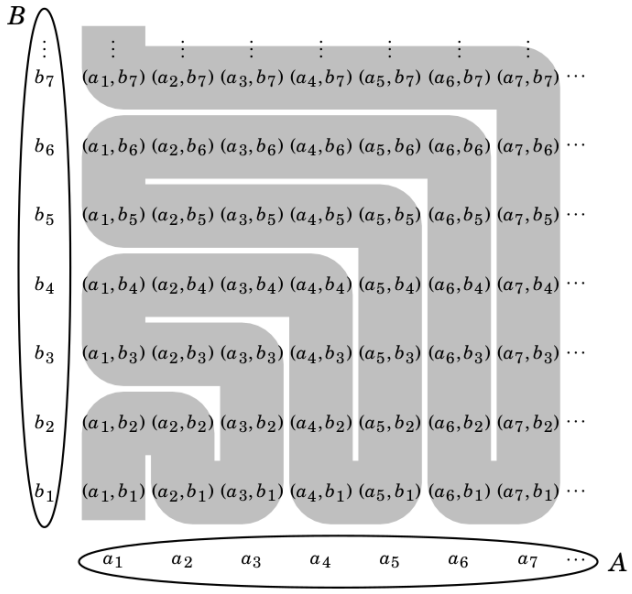


Théorème 19

Si A et B sont tout deux dénombrablement inifini, **alors** $A \times B$ l'est également.

Preuve 13

Il est possible de tracer un chemin impliquant toutes les paires ordonées de $A \times B$, pourvu que A et B son dénombrablement. Ce chemin représente la liste des éléments de l'ensemble $A \times B$, ce qui fait que $A \times B$ est également dénombrablement inifini. □



Note :

Le **corollaire** du théorème précédent est que si $A_1, A_2, A_3, \ldots, A_n$ avec $n \geq 2$, le produit cartésien $A_1 \times A_2 \times A_3 \times \cdots \times A_n$ *est également dénombrablement infini.*

Théorème 20

i A et B sont tous deux dénombrablement infinit, **alors** $A \cup B$ est dénombrablement infini.

Preuve 14

Supposons que A et B sont **tout deux dénombrablement infini**. Par le théorème précédent, on peut écrit A et B comme suit :

$$A = \{a_1, a_2, a_3, a_4 \cdots\}$$
$$B = \{b_1, b_2, b_3, b_4, \cdots\}$$

On peut alors arrangfer A et B dans **une seule** liste infinie comme suit

$$A \cup B = \{a_1, b_1, a_2, b_2, a_3, b_3, a_4, b_4, \cdots\}$$

Nous pouvons construire cette liste en nous imposant la restriction de ne pas lister un élément deux fois, s'il appartient aux deux liste. La liste que nous formons ainsi *liste* les éléments de l'ensemble $A \cup B$ et, ainsi, il s'ensuit que $A \cup B$ est dénombrablement infini.

7.2 Fonction incalculables §Hammack 18.4

Définition 42

Un fonction est $f : \mathbb{N} \to \mathbb{N}$ est dite **calculable** s'il existe une procédure prenant **n'importe que naturel** $n \in \mathbb{N}$ comme entré et sortant $f(n)$.

Note :

Les fonctions $f(n) = n!$ et $f(n) = \texttt{Fib}(n)$ sont calculables, puisqu'elles **possèdent une procédure** qui prend $n \in \mathbb{N}$ et rejette un $f(n)$. **Or**, la plupart des fonctions $f : \mathbb{N} \to \mathbb{N}$ *ne sont pas* **calculables**.

Proposition 7.1

L'ensemble de **toutes les fonctions** $f : \mathbb{N} \to \mathbb{N}$ **n'est pas** dénombrable (comptable).

Proposition 7.2

L'ensemble de **toutes les procédures possibles** est **dénombrablement** infini.

Proposition 7.3

Par la proposition précédente (7.2), l'ensemble de tous les programmes (ou procédures) qui peuvent être écrites dans un language de programmation donné est comptable.

Théorème 21

Il existe des fonctions qui ne sont pas calculables, c'est-à-dire qu'**elles ne peuvent pas** être calculé par une procédure (conséquence de 7.1, 7.2, 7.3).