

Deuxième année de Licence MIASHS

Mathématiques pour l'informatique¹

Julien GREPAT²

Contents

I Arithmétique	5
1 Rappels sur la division euclidienne	5
1.1 Entiers naturels	5
1.2 La division euclidienne	5
2 Numération	6
2.1 Définitions	6
2.2 Exemples	7
2.3 Notations et propriétés	7
2.4 Conversion	7
2.4.1 Exemple	8
2.4.2 Méthode	8
2.5 Numération des réels	9
2.6 Opérations	10
2.6.1 L'addition	10
2.6.2 La soustraction	10
2.6.3 Le produit	10
3 Multiples et diviseur	11
3.1 Multiples	11

¹Reproduction et diffusion interdite sans l'accord de l'auteur

²Contact : julien.grepas@gmail.com

3.2	diviseurs	11
4	Nombre premiers	12
4.1	Définition	12
4.2	Caractérisation	12
4.3	Décomposition en nombres premiers	12
5	PGCD et PPCM	13
5.1	Définition	13
5.2	algorithme d'Euclide	13
5.3	Nombres premiers entre eux	13
5.4	PGCD et décomposition en nombres premiers	14
5.5	PPCM et décomposition en nombres premiers	14
5.6	Propriétés sur le PGCD et le PPCM	14
5.7	Égalité de BACHET–BEZOUT	15
5.8	Algorithme d'Euclide généralisé	15
6	Congruences	16
6.1	Définition	16
6.2	Propriétés	17
6.3	Petit théorème de Fermat	17
6.4	Groupe cyclique	18
6.5	Théorème chinois des restes	19
6.6	Codage RSA	20
6.6.1	Scénario	20
6.6.2	Déchiffrement RSA	20
6.6.3	Exemple	21
II	Logique mathématiques	23
7	Propositions	23
7.1	Les connecteurs logiques	23
7.2	Règles de calcul propositionnel	24
7.3	Clauses et formes normales	25
7.4	Modus ponens – Principe de résolution de Robinson	27
8	Prédicats, quantificateurs	27

8.1	Quantificateurs	27
8.1.1	Ordre des quantificateurs	28
8.1.2	Négation	28
8.1.3	Ensembles et logique : Diagrammes de Venn	29
8.1.4	Skolémisation	30
9	Algèbre de Boole	31
9.1	Lien avec la logique mathématique	31
9.1.1	Loi de Morgan	32
9.2	Tableau de Karnaugh	32
9.2.1	Cas de deux variables	32
9.2.2	Cas de trois variables	33
9.2.3	Cas de quatre variables	34
10	Récurrence et récursivité	35
10.1	Preuve par récurrence	35
10.2	Mise en place d'un raisonnement récursif en mathématiques	35
10.3	Définitions mathématiques par récursion	36
10.4	Des preuves par récursion ?	36
10.5	Algorithme itératif	37
10.5.1	Boucle <code>for</code>	37
10.5.2	Boucle <code>while</code>	37
10.5.3	La factorielle	37
10.6	Algorithme récursif	38
10.7	Algorithme itératifs Vs récursif	38
III	Théorie des graphes	39
11	Introduction	39
12	Matrice d'adjacence	40
12.1	Définitions	40
12.2	Chemin et longueur dans un graphe orienté	42
12.3	Nombre de chemins de longueur donnée	43
12.4	Fermeture transitive	43
13	Algorithme de Dijkstra	44

13.1	Graphe non orienté	44
13.2	Arbre	46
13.3	Graphe pondéré (ou valué)	46
13.4	Recherche de chemin le plus court – Algorithme de Dijkstra	47
13.4.1	l'Algorithme	47
13.4.2	Exemple : le tram de Grenoble	50
14	Dessin d'un graphe par niveaux	51
15	Ordonnancement	52
15.1	Date au plus tôt	54
15.2	Date au plus tard	55
15.3	Marges d'une tâche	56

Part I

Arithmétique

1 Rappels sur la division euclidienne

Nous rappelons ici quelques définitions sur les entiers et la division euclidienne indispensables à la manipulation efficace de l'arithmétique.

1.1 Entiers naturels

Les entiers naturels sont $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, \dots$. Cet ensemble est noté \mathbb{N} . Les entiers relatifs sont quand à eux signés, $\dots, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, \dots$. Cet ensemble est noté \mathbb{Z} .

1.2 La division euclidienne

Effectuer la division euclidienne d'un entier naturel A par un entier naturel B non nul c'est déterminer les uniques entiers Q (appelé quotient) et R (appelé reste) tels que :

$$A = BQ + R,$$

et $0 \leq R < B$.

Exemple. Le quotient et le reste de la division euclidienne de $A = 53$ par $B = 6$ sont respectivement $Q = 8$ et $R = 5$. En effet,

$$53 = 6 \times 8 + 5$$

et $0 \leq 5 < 6$.

Rappelons-nous de comment on posait une division lorsqu'on était petit... On souhaite poser la division euclidienne de 178 par 3.

	<i>Étape 1 : on forme le plus petit nombre supérieur au diviseur dans le dividende, en partant de la gauche.</i> <i>Étape 2 : dans 17, on peut faire 5 paquets de 3 : on note 5 au quotient, on soustrait $3 \times 5 = 15$ à 17 / on garde donc 2.</i> <i>Étape 3 : on abaisse le chiffre d'après. Dans 28, il y a 9 paquets de 3 - On note 9 au quotient, on soustrait $9 \times 3 = 27$ à 28. Il reste 1</i> ⚠ Stop, on ne passe pas à droite de la virgule !
--	--

On vérifie alors que

$$178 = 3 \times 59 + 1,$$

avec $1 < 3$.

2 Numération

2.1 Définitions

Trois numérations nous sont indispensables dans ce que nous souhaitons faire.

- L'être humain compte naturellement en base 10 avec les dix chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 :
 $10, 11, 12, \dots, 97, 98, 99, 100, 101, 102, \text{ etc.}$
- Le microprocesseur d'un ordinateur ne travaille qu'avec deux chiffres : 0 (pas de courant) et 1 (courant). Les calculs s'y font donc naturellement en base 2 (on n'utilise que les chiffres 0

et 1) :

0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, etc.

- Pour compter en base 16, on utilise les dix chiffres et on en rajoute six autres (que l'on note A, B, C, D, E et F). Cela donne :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C,
1D, 1E, 1F, 20, 21, 22, etc.

2.2 Exemples

- 4 en base 10, c'est 100 en base 2.
- 9 en base 10, c'est 1001 en base 2.
- 1A en base 16, c'est 26 en base 10.
- A en base 16, c'est 10 en base 10.

2.3 Notations et propriétés

Le nombre 23, écrit en base 10, se note 10111 en base 2 et 17 en base 16, mais on ne peut pas écrire $23 = 10111 = 17$. C'est pourquoi nous adoptons la notation $(n)_p$ pour indiquer que le nombre n est écrit en base p :

$$(23)_{10} = (10111)_2 = (17)_{16}.$$

Les écritures en bases 2, 10 et 16 s'appellent aussi respectivement les écritures binaire, décimale et hexadécimale. Lorsqu'un nombre est écrit en base 10, on peut simplifier la notation $(x)_{10}$ en x . Par exemple $(201)_{10}$ peut s'écrire simplement 201.

Proposition 2.1 *L'écriture d'un nombre entier en base 2, 10 ou 16 est unique.*

2.4 Conversion

Pour convertir un entier d'une base à l'autre, il est important de comprendre la relation algébrique que l'on a entre une base p et l'écriture du nombre dans cette base p . C'est ce qu'énonce la propriété suivante.

Proposition 2.2 *Quels que soient les nombres entiers $a_0, a_1, a_2, \dots, a_n$ compris entre 0 et $p - 1$, où p désigne 2, 10 ou 16, on a*

$$(a_n \cdots a_2 a_1 a_0)_p = a_n \times p^n + \cdots + a_2 \times p^2 + a_1 \times p + a_0.$$

Dans le cas où $p = 16$, on remplace dans l'écriture du membre de gauche, tout a_i égal à 10, 11, 12, 13, 14 ou 15 par A, B, C, D, E ou F respectivement.

2.4.1 Exemple

Conversion vers la base 10 :

$$(11011)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 = 16 + 8 + 2 + 1 = 27$$

$$(5C8)_{16} = 5 \times 16^2 + 12 \times 16 + 8 = 1280 + 192 + 8 = 1480$$

2.4.2 Méthode

Il existe plusieurs méthodes pour convertir un entier vers la base $p = 2$ ou 16 .

- Nous allons déterminer l'écriture en base 2 du nombre 75.
- Nous déterminerons alors l'écriture en base 16 du nombre 2 014 (méthodes 1 et 2).
- Nous effectuerons des conversions directes entre les bases 2 et 16 (méthode 3).

• Méthode 1

Handwritten notes for Method 1:

75 = $\overbrace{1 \times 2^6 + 11}^{64}$

11 = $1 \times 2^3 + 3$

3 = $1 \times 2 + 1$

1

On effectue la division euclidienne du nombre par la puissance de p qui lui est inférieure ou égale puis on recommence sur le reste.

Donc $75 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2 + 1$

$= 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1$

$= (1001011)_2$

De même:

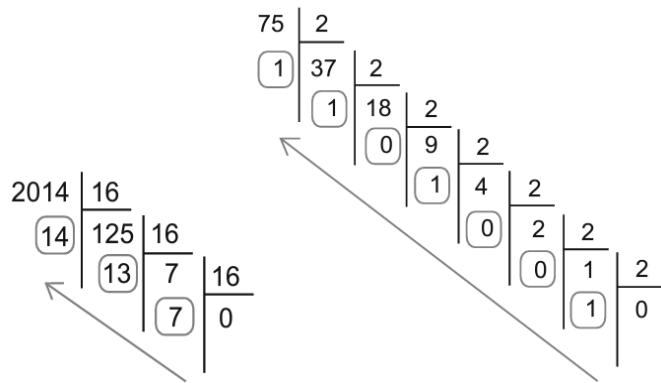
$2014 = 7 \times 16^2 + 222$

$222 = 13 \times 16 + 14$

$2014 = 7 \times 16^2 + 13 \times 16 + 14$

$= (7DE)_{16}$

- **Méthode 2** On effectue la division euclidienne du nombre par p puis on recommence avec le quotient et ainsi de suite jusqu'à obtenir 0. À la fin, on inverse l'ordre des restes obtenus.



Ainsi, on retrouve les résultats précédents :

$$75 = (1001011)_{10}, \quad 2014 = (7DE)_{16}.$$

- **Méthode 3** on peut passer directement de la base 2 à la base 16, et inversement, en utilisant le tableau de conversion suivant :

base 16	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
base 2	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

- Pour la conversion directe de $(2B)_{16}$ en base 2, on concatène, à partir du tableau, les écritures de 2 et de B en base 2, puis on supprime les zéros inutiles :

$$(2B)_{16} = (00101011)_2 = (101011)_2.$$

- Pour la conversion directe de $(110111)_2$ en base 16, on ajoute à gauche des zéros inutiles pour faire des paquets de quatre chiffres que l'on remplace ensuite par leurs équivalents en base 16.

$$(110111) - 2 = (\underbrace{0011}_2 \underbrace{0111}_2)_2 = (37)_{16}.$$

2.5 Numération des réels

Dans le dernier paragraphe, nous avons défini les écritures binaire, décimale et hexadécimale d'un nombre entier. Plus largement, tout nombre réel peut être écrit en base 2, 10 ou 16. Pour comprendre, prenons l'exemple de deux réels et de la base 10.

$$(53,627)_{10} = 5 \times 10 + 3 + 6/10 + 2/10^2 + 7/10^3,$$

et

$$(456,905)_{10} = 4 \times 10^2 + 5 \times 10 + 6 + 9/10 + 0/10^2 + 5/10^3.$$

Plus généralement on a la propriété suivante.

Proposition 2.3 *Quels que soient les nombres entiers a_0, \dots, a_n et b_1, \dots, b_m compris entre 0 et $p-1$ (où p désigne 2, 10 ou 16), on a :*

$$(a_n \cdots a_1 a_0, b_1 \cdots b_m)_p = a_n^n + \cdots + a_1 p + a_0 + \frac{b_1}{p} + \cdots + \frac{b_m}{p^m}.$$

Par exemple :

$$7,25 = 7 + 0,25 = 1 \times 2^2 + 1 \times 2 + 1 + \frac{1}{2^2} = (111,01)_2.$$

Bon courage pour faire des conversions à la main !

2.6 Opérations

En base 2, 10 ou 16, on fait les opérations élémentaires de la même façon.

2.6.1 L'addition

$$\begin{array}{r}
 \begin{array}{r}
 1 & 1 \\
 & 1 & 0 & 1 \\
 + & 1 & 1 & 1 & 0 \\
 \hline
 1 & 0 & 0 & 1 & 1
 \end{array}
 \left| \begin{array}{l}
 (1)_2 + (0)_2 = (1)_2 \\
 (0)_2 + (1)_2 = (1)_2 \\
 (1)_2 + (1)_2 = (10)_2 : \text{on pose } 0, \text{on retient } 1. \\
 (1)_2 + (1)_2 = (10)_2 : \text{on pose } 0, \text{on retient } 1.
 \end{array} \right.
 \end{array}$$

2.6.2 La soustraction

$$\begin{array}{r}
 \begin{array}{r}
 1 & 1 & 1 & 1 \\
 - & 1 & 0 & 1 & 1 & 0 \\
 & 1 & 1 & 1 \\
 \hline
 1 & 1 & 1 & 1 \\
 0 & 1 & 1 & 1 & 1
 \end{array}
 \left| \begin{array}{l}
 \text{Puisque } 1 > 0, \text{on pose une retenue de } 1, \\
 \text{puis } (10)_2 - (1)_2 = (1)_2. \\
 \text{Nouvelle retenue de } 1 \text{ puis } (11)_2 - ((1)_2 + (1)_2) = (1)_2, \text{ etc.}
 \end{array} \right.
 \end{array}$$

2.6.3 Le produit

$$\begin{array}{r}
 \begin{array}{r}
 F & 3 \\
 \times & 2 & A \\
 \hline
 9 & 7 & E \\
 + & 1 & E & 6 & . \\
 \hline
 2 & 7 & D & E
 \end{array}
 \left| \begin{array}{l}
 (A)_{16} \times (3)_{16} = 10 \times 3 = 30 = (1E)_{16}, \text{on pose } E, \text{on retient } 1. \\
 (A)_{16} \times (F)_{16} = 10 \times 15 = 150 = (96)_{16}, \text{avec la retenue, on obtient,} \\
 (96)_{16} + (1)_{16} = (97)_{16}, \text{d'où } 97E. \\
 (2)_{16} \times (3)_{16} = (6)_{16}, \text{on pose } 6. \\
 (2)_{16} \times (F)_{16} = 2 \times 15 = 30 = (1E)_{16}, \text{on obtient } 1E6 \text{ avec le décalage} \\
 \text{« . ».} \\
 \text{On effectue ensuite l'addition } (97E)_{16} + (1E60)_{16} : (E)_{16} + (0)_{16} = (E)_{16}, \\
 \text{on pose } E ; (7)_{16} + (6)_{16} = 7 + 6 = 13 = (D)_{16}, \text{on pose } D ; (9)_{16} + (E)_{16} \\
 = 9 + 14 = 23 = (17)_{16}, \text{on pose } 7 \text{ et on retient } 1 ; \text{enfin, } (1)_{16} + (1)_{16} \\
 = (2)_{16}, \text{on obtient alors le résultat } 27DE.
 \end{array} \right.
 \end{array}$$

Bon courage pour la division euclidienne !

3 Multiples et diviseur

3.1 Multiples

Definition 3.1 Soit a un nombre entier. On appelle multiple de a tout nombre qui s'écrit $a \times b$ où b est un nombre entier.

Par exemple, $30 = 3 \times 10$ est un multiple de 3.

3.2 diviseurs

Definition 3.2 Soit a un nombre entier. Le nombre entier b est un diviseur de a s'il n'y a pas de reste dans la division (entièrre) de a par b .

On pourra présenter les diviseurs par couple dans une liste ordonnée croissante :

Diviseurs de 84						
1	2	3	4	6	7	
84	42	28	21	14	12	→ le suivant devrait 11

Ainsi, lorsqu'on retrouve un diviseur déjà donné, c'est que la liste est complète.

Proposition 3.3 Nous avons les propriétés suivantes.

- Si a divise b , alors a divise tout multiple de b .
- Si a divise b et b divise c alors a divise c .
- Si a et b sont divisibles par c , alors c divise

$$au + bv, \quad \forall u, v \in \mathbb{Z}.$$

En particulier, c divise $a + b$ et $a - b$.

Note. Souvenirs de sixième...

- * Un nombre est divisible par 3 si la somme de ses chiffres est divisible par 3.
 - * Un nombre est divisible par 4 si le nombre formé par les dizaines et les unités est un multiple de 4
- $810\ 116 \rightarrow$ multiple de 4
- 810 116 est divisible par 4 -
- * Un nombre est divisible par 5 s'il se termine par 0 ou 5
 - * Un nombre est divisible par 9 si la somme de ses chiffres est divisible par 9
 - * Un nombre est divisible par 10 s'il se termine par 0.

4 Nombre premiers

4.1 Définition

Definition 4.1 Un nombre premier est un nombre qui n'a pour diviseur que 1 et lui-même.

Les premiers nombres premiers sont 2, 3, 5, 7, 11, 13, 17. Mise à part 2 ils sont tous impairs.

4.2 Caractérisation

Theorem 4.2 Soit n un entier naturel. Si n n'est pas premier, il admet au moins un diviseur premier inférieur ou égal à \sqrt{n} .

Il s'agit d'un très bon critère pour savoir si un nombre est premier. On peut chercher si 347 est un nombre premier. On cherche si les nombres premiers inférieurs à $\sqrt{347} = 18,6$ divisent 347. Il s'agit de {2, 3, 5, 7, 11, 13, 17}. Aucun d'entre eux ne divise 347. Ce nombre est premier.

4.3 Décomposition en nombres premiers

Proposition 4.3 Tout nombre entier peut se décomposer en produit de facteurs premiers. Cette décomposition est unique.

Par exemple, $70 = 2 \times 5 \times 7$ est la décomposition en facteurs premiers de 70.

5 PGCD et PPCM

5.1 Définition

Definition 5.1 On appelle PPCM le plus petit multiple commun de deux nombres.

Par exemple, 15 et 10 ont pour PPCM le nombre 30.

Definition 5.2 On appelle PGCD le plus grand diviseur de deux nombres.

Par exemple, 9 et 15 ont pour PGCD le nombre 3.

5.2 algorithme d'Euclide

On pourra utiliser l'algorithme d'Euclide pour calculer le PGCD. Cherchons, par exemple, le PGCD de 495 et 210.

The image shows handwritten calculations for the Euclidean algorithm:

$$\begin{aligned} 495 &= 210 \times 2 + 75 \\ 210 &= 75 \times 2 + 60 \\ 75 &= 60 \times 1 + 15 \\ 60 &= 15 \times 4 + 0 \end{aligned}$$

Annotations in blue and red explain the process:

- "on pose la division par le reste" (we perform the division by the remainder) is written next to the first equation.
- "Et ainsi de suite" (and so on) is written next to the second equation.
- "On garde le dernier reste non nul." (We keep the last non-zero remainder) is written next to the fourth equation.

Ainsi $\text{PGCD}(495, 210) = 15$. Cet algorithme se justifie par la propriété suivante :

Proposition 5.3 Soit a et b des entiers naturels, $a > b$. En posant la division euclidienne $a = bq+r$, on a $\text{PGCD}(a, b) = \text{PGCD}(b, r)$.

5.3 Nombres premiers entre eux

Definition 5.4 Deux nombres entiers a et b sont dits premiers entre eux si leur seul diviseur commun est 1.

Il vient alors immédiatement que si a et b sont premiers entre eux, alors $\text{PGCD}(a, b) = 1$.

5.4 PGCD et décomposition en nombres premiers

Soient a et b deux entiers naturels supérieurs ou égaux à 2, et leur décomposition en facteurs premiers.

- S'ils n'ont aucun facteur commun, leur PGCD vaut 1, ils sont premiers entre eux.
- Leur PGCD est le produit des facteurs communs aux deux décompositions, chaque facteur étant affecté avec la plus petite puissance des deux décomposition.

Par exemple, $a = 4950 = 2 \times 3^2 \times 5^2 \times 11$ et $b = 4875 = 3 \times 5^3 \times 13$. On a alors

$$PGCD(4950, 4875) = 3 \times 5^2 = 75.$$

5.5 PPCM et décomposition en nombres premiers

Soient a et b deux entiers naturels supérieurs ou égaux à 2, et leur décomposition en facteurs premiers.

- S'ils n'ont que des facteurs communs, ils sont égaux, leur PPCM est eux-même.
- Leur PPCM est le produit de tous les nombres premiers qui apparaissent dans au moins une des décompositions en facteurs premiers de ces deux entiers, chacun affecté du plus grand exposant qui apparaît dans celles-ci.

Par exemple, $a = 4950 = 2 \times 3^2 \times 5^2 \times 11$ et $b = 4875 = 3 \times 5^3 \times 13$. On a alors

$$PPCM(4950, 4875) = 2 \times 3^2 \times 5^3 \times 11 \times 13.$$

5.6 Propriétés sur le PGCD et le PPCM

Des deux sections précédentes, on déduit assez rapidement les propriétés suivantes.

Proposition 5.5 *Soit a, b des nombres entiers supérieurs à 2.*

- *Les diviseurs communs à deux nombres sont les diviseurs de leur PGCD.*
- $PGCD(ca, cb) = cPGCD(a, b).$
- *Si c est un diviseur commun de a et b ,*

$$PGCD(a/c, b/c) = PGCD(a, b)/c.$$

- $PGCD(a, b)|PGCD(ac, bd).$
- *Si c est un diviseur commun de a et b , $c = PGCD(a, b)$ si et seulement si a/c et b/c sont premiers entre eux.*

- Soient a et b deux entiers naturels supérieurs ou égaux à 2, on a

$$PGCD(a, b) \times PPCM(a, b) = ab.$$

On a également les critères de divisibilité suivants. Des deux sections précédentes, on déduit assez rapidement les propriétés suivantes.

Proposition 5.6 Soit a, b des nombres entiers supérieurs à 2 premiers entre eux.

- Si a divise bc , alors a divise c .
- On a $PGCD(a, bc) = PGCD(a, c)$.
- Si de plus a et c sont premiers entre eux, alors $PGCD(a, bc) = 1$.

De ces propriétés découle le résultat suivant.

Proposition 5.7 Si p est un nombre premier et s'il divise un produit, alors il divise au moins un des termes.

5.7 Égalité de BACHET–BEZOUT

Les deux résultats suivants, nommés respectivement Théorème de BACHET–BEZOUT et Théorème de BEZOUT, concluent cette section et ont des applications fondamentales en cryptographie.

Theorem 5.8

- Soient a et b deux entiers relatifs. Si d est le PGCD de a et b , alors il existe deux entiers relatifs x et y tels que

$$ax + by = d.$$

- Deux entiers relatifs a et b sont premiers entre eux si et seulement s'il existe deux entiers relatifs x et y tels que

$$ax + by = 1.$$

5.8 Algorithme d'Euclide généralisé

On peut déterminer les coefficients dans le théorème 5.8 en “remontant” l'algorithme d'Euclide. Pour mémoire :

$$\begin{aligned}
 495 &= 210 \times 2 + 75 \\
 210 &= 75 \times 2 + 60 \\
 75 &= 60 \times 1 + 15 \\
 60 &= 15 \times 4 + 0
 \end{aligned}$$

on pose la division par le reste
 Et ainsi de suite
 On garde le dernier reste non nul.

En remontant cet algorithme, on a

Algorithme d'Euclide $15 = 3 \times 495 - 7 \times 210$
 \Downarrow
 $15 = -210 + 3(495 - 2 \times 210)$

$$\begin{aligned}
 210 &= 75 \times 2 + 60 & 15 &= 75 - (210 - 2 \times 75) = -210 + 3 \times 75 \\
 75 &= 60 \times 1 + 15 & 15 &= 75 - 60 \\
 60 &= 15 \times 4 + 0 & \uparrow &
 \end{aligned}$$

\Downarrow

$$\text{PGCD}(495, 210) = 15.$$

on obtient les coefficients du théorème 5.8

$$3 \times 495 - 7 \times 210 = 15.$$

6 Congruences

6.1 Définition

Definition 6.1 Soit n un entier naturel. On dit que les entiers a et b sont congrus modulo n si et seulement si ils ont le même reste dans la division euclidienne par n .

On note alors

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

Il est immédiat de voir que si

$$a = nq + r,$$

alors

$$a \equiv r[n].$$

C'est ce reste r qui représentera l'entier a modulo n .

Par exemple, puisque

$$101 = 7 \times 14 + 3,$$

on a

$$101 \equiv 3[7].$$

Puisque $66 = 7 \times 9 + 3$, $66 \equiv 3[7]$. On peut aussi écrire $66 \equiv 101[7]$.

6.2 Propriétés

Les congruences rendent élémentaire le calcul du reste. Tout est dans la proposition suivante.

Proposition 6.2 *Soient a, b, c et d trois nombres entiers naturels et n un entier naturel non nul. Si $a \equiv c[n]$, et $b \equiv d[n]$,*

$$\begin{aligned} a + b &\equiv c + d[n], \\ a - b &\equiv c - d[n], \\ ab &\equiv cd[n], \\ a^p &\equiv c^p[n], \end{aligned}$$

pour tout entier naturel p .

Par exemple, on rappelle que $101 \equiv 3[7]$ et que $66 \equiv 3[7]$. Ainsi :

$$\begin{aligned} 167 &= 101 + 66 \equiv 6[7], \\ 35 &= 101 - 66 \equiv 0[7], \\ 6666 &= 66 \times 101 \equiv 3 \times 3[7] \equiv 9[7] \equiv 2[7], \\ 10201 &= 101^2 \equiv 3^2[7] \equiv 2[7]. \end{aligned}$$

On notera que si $a \equiv b[n]$ alors $a - b \equiv 0[n]$ et n divise $a - b$.

6.3 Petit théorème de Fermat

Le théorème qui suit est fondamental dans la cryptologie moderne industrielle qui a besoin d'entiers naturels premiers à grands nombres de chiffres (codage RSA).

Il dit que si p est un nombre premier et si a est un entier non divisible par p , alors $a^{p-1} - 1$ est un multiple de p . Un énoncé équivalent est : "si p est un nombre premier et si a est un entier quelconque, alors $a^p - a$ est un multiple de p ".

Theorem 6.3 Si p est un nombre premier et si a est un entier non divisible par p , alors

- $a^{p-1} \equiv 1[p]$,
- ou, de manière équivalente, $a^p \equiv a[p]$.

On a pour application directe

$$2^{97} - 2 = 158456325028528675187087900670$$

est divisible par 97.

Corollary 6.4 Soit p et q des nombres premiers distincts. On pose $n = pq$. Pour tout a premier avec n on a

$$a^{(p-1)(q-1)} \equiv 1[n].$$

Proof. Notons que si $a \equiv b[p]$ et $a \equiv b[q]$ (avec p et q premiers distincts) alors $a \equiv b[pq]$. En effet, on a

$$a - b = kp, \quad a - b = mq.$$

Donc p divise mq , mais p est premier avec q , donc p divise m . Donc $a - b$ est un multiple de pq , et $a \equiv b[pq]$.

Il reste à voir que, par le petit théorème de Fermat,

$$a^{p-1} \equiv 1[p], \quad a^{q-1} \equiv 1[q].$$

Il suit que

$$a^{(p-1)(q-1)} \equiv 1[pq] \equiv 1[n].$$

□

6.4 Groupe cyclique

On s'aperçoit que, pour n fixé, les congruences reviennent à considérer un nombre comme son reste dans la division par n . Par conséquent, les nombres

$$p, \quad p + n, \quad p - n, \quad p + 2n, \quad p - 2n, \quad p + 3n, \quad \dots$$

ont tous le même reste $p < n$ dans la division par n . Ils ont donc la même congruence modulo n , qui est p . Cette classe d'équivalence \bar{p} représente donc tous ces nombres.

L'ensemble des classes d'équivalences

$$\bar{0}, \bar{1}, \dots, \overline{n-2}, \overline{n-1},$$

forment l'ensemble $\mathbb{Z}/n\mathbb{Z}$. On définit une somme et un produit entre classes d'équivalences grâce à la propriété 6.2. Avec la stabilité pour ces deux lois, la structure de $\mathbb{Z}/n\mathbb{Z}$ s'appelle groupe. Le groupe est dit cyclique car en ajoutant 1 à \bar{a} , et en itérant l'implémentation, on finira par revenir sur \bar{a} au bout d'un certain nombre de fois.

Definition 6.5

- On peut définir l'inverse \bar{b} d'un élément \bar{a} comme la classe d'équivalence telle que

$$\bar{a}\bar{b} \equiv 1[n].$$

- Soit $\bar{a} \in \mathbb{Z}/n\mathbb{Z}$, s'il existe $\bar{b} \in \mathbb{Z}/n\mathbb{Z}$ telle que $\bar{a}\bar{b} = \bar{0}$, on dit que \bar{a} est un diviseur de zéro.

On a les résultats suivants.

Theorem 6.6

- Soit $\bar{a} \in \mathbb{Z}/n\mathbb{Z}$, \bar{a} est inversible si et seulement si $\text{PGCD}(a, n) = 1$.
- Si n est premier, seule $\bar{0}$ n'est pas inversible.
- Soit $\bar{a} \in \mathbb{Z}/n\mathbb{Z}$, \bar{a} est un diviseur de zéro si a et n ne sont pas premiers entre eux.

6.5 Théorème chinois des restes

Le théorème chinois des restes, dont le nom renseigne sur l'endroit de première parution, nous montre comment décomposer un ensemble $\mathbb{Z}/n\mathbb{Z}$ sur deux (ou plusieurs) espaces plus simples grâce à la décomposition en facteurs premiers.

Theorem 6.7 Soit $n = n_1 n_2$ avec les nombres n_1 et n_2 premiers entre eux. On note u_1 et u_2 tels que

$$u_1 n_1 + u_2 n_2 = 1,$$

dont l'existence est assurée par le théorème de Bachet-Bezout 5.8. Pour $x \in \mathbb{Z}$,

$$\begin{cases} x \equiv r_1 [n_1] \\ x \equiv r_2 [n_2] \end{cases} \iff x \equiv r_2(u_1 n_1) + r_1(u_2 n_2)[n].$$

En pratique, cela signifie que l'on peut décomposer $\mathbb{Z}/12\mathbb{Z}$ sur $\mathbb{Z}/3\mathbb{Z}$ et $\mathbb{Z}/4\mathbb{Z}$. On a

$$1 \times 4 - 1 \times 3 = 1.$$

Par conséquent,

$\mathbb{Z}/12\mathbb{Z}$	0	1	2	3	4	5	6	7	8	9	10	11
$\mathbb{Z}/3\mathbb{Z}$	0	1	2	0	1	2	0	1	2	0	1	2
$\mathbb{Z}/4\mathbb{Z}$	0	1	2	3	0	1	2	3	0	1	2	3

évident

Théorème Chinois

C'est évident.
Si $x \equiv 5[12]$: $5 \equiv 2[3]$
et $5 \equiv 1[4]$

Réciproquement, si
 $x \equiv 2[4]$ et $x \equiv 1[3]$
alors $x \equiv 1 \times 1 \times 4 + 2 \times (-1) \times 3[12]$
 $1 \times 4 - 1 \times 3 = 1$

Il vient que

$$x \equiv 4 - 6[12] \equiv -2[12] \equiv 10[12].$$

6.6 Codage RSA

Il s'agit d'un codage très populaire, utilisé à une échelle industrielle, sur lequel ont travaillé Rivest, Shamir et Adleman, MIT (USA). R et S sont informaticiens, A est un matheux.

L'idée de RSA est d'utiliser la difficulté de décomposer un nombre en produit de (grand) facteurs premiers. Bref, la méthode est publique, une des clés est publique, les données cryptées sont publiques, mais sans la clé privée, personne ne peut rien faire (attention à l'ordinateur quantique).

6.6.1 Scénario

Alice souhaite que Bob lui envoie un message m . Elle envoie à Bob la clef publique pour chiffrer le message. Ainsi, Alice pourra le déchiffrer grâce à sa clef privée. Le principe de chiffrement est connu tous, les clefs publiques sont connues de tous. On notera e (comme encoding) la clef publique, et d (comme decoding) la clef privée.

Soit m le nombre à chiffrer par Bob, et x le nombre chiffré envoyé par Bob.

Choisissons $n = pq$ où p et q sont des grands nombres premiers distincts. Il est à noter que, connaissant n , il est très difficile de trouver p et q .

Bob va envoyer x tel que

$$x \equiv m^e[n].$$

Alice déchiffrera

$$m \equiv x^d[n].$$

Il reste à choisir d, e , à rendre public (à Bob) la clef (n, e) .

6.6.2 Déchiffrement RSA

La possibilité de chiffrer-déchiffrer réside dans le résultat suivant.

Lemma 6.8 Soit $n = pq$ où p et q sont des nombres premiers distincts. On pose $\varphi(n) = (p-1)(q-1)$. On se donne e premier avec $\varphi(n)$. Soit d un inverse de e modulo $\varphi(n)$. Alors, pour tout entier m ,

$$x \equiv m^e[n] \quad \Rightarrow \quad m \equiv x^d[n].$$

Note. On peut utiliser l'algorithme d'Euclide généralisé pour trouver d et e .

Proof. Montrons que $m \equiv m^{ed}[n]$.

- Si m est premier avec n , d'après le corolaire 6.4, puisque $ed \equiv 1[\varphi(n)]$,

$$m^{ed} \equiv m^{1+k\varphi(n)}[n] \equiv m \times 1[n].$$

- Supposons, sans restriction, que $0 \leq m < n$. Si m n'est pas premier avec $n = pq$, alors ils ont un facteur premier commun (pas les deux puisque $m < pq$). Disons arbitrairement p . Ainsi m et q sont premiers entre eux. Notons que $m^{ed} \equiv 0[p] \equiv m$ et, comme précédemment, $m^{ed} \equiv m^{1+k\varphi(n)}[q] \equiv m(m^{(q-1)})^{k(p-1)}[q] \equiv m \times 1^{k(p-1)}[q]$, d'après le petit théorème de Fermat. On a alors

$$m^{ed} - m = ap, \quad m^{ed} - m = bq.$$

Donc p divise bq , mais p est premier avec q , donc p divise b . Donc $a - b$ est un multiple de pq , et $m^{ed} \equiv m[pq]$.

□

Note. En pratique, c'est une **portion** de message à transmettre qui est d'abord transformée en un nombre x . Plusieurs lettres doivent être groupées pour éviter l'analyse fréquentielle.

6.6.3 Exemple

- (i) Alice prépare ses clés pour elle et pour Bob.

- Elle choisit $p = 11$ et $q = 23$, d'où $n = 253$. On calcule aussi $\varphi(n) = (p-1)(q-1) = 10 \times 22 = 220$.
- Elle choisit e un nombre premier avec $\varphi(n) = 220$. Par exemple $e = 3$.
- Elle calcule l'inverse d de e modulo $\varphi(n) = 220$:

$$ed \equiv 1[220] \quad \iff \quad 3d \equiv 1[220] \quad \iff \quad 3d - 220k = 1.$$

Utilisons l'algorithme d'Euclide généralisé.

$$220 = 3 \times 73 + 1 \quad \iff \quad 220 - 3 \times 73 = 1.$$

Notons que -73 n'est pas top, et que sa classe d'équivalence est $-73 \equiv 220 - 73[220] \equiv 147[220]$. On gardera $d = 147$.

- envoie la clef publique $(253, 3)$ et garde sa clé $(253, 147)$.

(ii) Bob crypte son message $m = 123$ par exemple. Il crypte

$$x \equiv 123^3 [253] \equiv 52[253].$$

Bob envoie $x = 52$.

(iii) Alice déchiffre

$$m \equiv 52^{147} [253] \equiv 123[253],$$

merci Python ! Alice a déchiffré le bon nombre !

Part II

Logique mathématiques

On définit les notions suivantes :

- une **proposition** est un énoncé qui est soit vrai, soit faux ;
- Vrai (1) ou Faux (0) sont appelés **valeurs de vérité** ;
- un **prédictat** est un énoncé contenant une ou plusieurs variables et qui se transforme en proposition suivant la valeur de ces variables.
- un atome est une proposition ou un prédictat élémentaire qui ne dépend pas d'autre proposition ou prédictat.

Exemple. Par exemple les énoncés suivants sont des assertions : $2 < 15$ (elle est vraie), $\sqrt{2}$ est un nombre rationnel (elle est fausse).

On considère la variable x et le prédictat $x + 6 = 10$. Pour $x = 4$, cette proposition est vraie, elle est fausse sinon.

Ce sont des atomes. Ils ne dépendent pas de plusieurs propositions ou prédictats, contrairement à la proposition suivante :

“les deux ascenseurs fonctionnent”,

qui est équivalente à

“l'ascenseur 1 fonctionne” et “l'ascenseur 2 fonctionne”.

Definition 6.9 Deux propositions sont dites logiquement équivalentes, ou plus simplement équivalentes, si elles sont toutes deux vraies ou toutes deux fausses.

7 Propositions

7.1 Les connecteurs logiques

L'élaboration de nouvelles assertions à partir d'autres se fait en utilisant les connecteurs logiques de négation, de conjonction, de disjonction, d'implication et d'équivalence. Dans ce qui suit, P et Q désignent des assertions.

- La négation de P , notée $\neg P$, ou non P ou \overline{P} , est l'assertion qui est vraie si P est fausse et fausse si P est vraie.

Par exemple la négation de l'assertion : “ x est strictement positif ” est “ x est négatif ou nul ” .

En théorie des ensembles on admet qu'il n'existe pas d'assertion P telle que P et \overline{P} soient toutes deux vraies. On dit que cette théorie est “non contradictoire ” .

Definition 7.1 *Un littéral est un atome ou sa négation.*

- La conjonction de P et Q , notée $P \wedge Q$ (lire P et Q), est l'assertion qui est vraie uniquement si P et Q sont toutes deux vraies (et donc fausse dans les trois autres cas).

Par exemple $P \wedge \overline{P}$ est toujours faux (on se place dans des théories non contradictoires).

- La disjonction de P et Q , notée $P \vee Q$ (lire P ou Q), est l'assertion qui est vraie uniquement si l'une des deux assertions P ou Q est vraie (donc fausse si P et Q sont toutes deux fausses). Il faut remarquer que le “ou” pour “ou bien” est inclusif, c'est-à-dire que P et Q peuvent être toutes deux vrais dans le cas où $P \vee Q$ est vraie.

On peut aussi introduire le “ou exclusif”, noté *xor* ou \oplus , qui est vrai uniquement lorsque l'une des deux assertions, mais pas les deux simultanément, est vraie.

- L'implication, notée $P \rightarrow Q$, est l'assertion qui est fausse uniquement si P est vraie et Q fausse (donc vraie dans les trois autres cas). On peut remarquer que si P est fausse, alors $P \rightarrow Q$ est vraie indépendamment de la valeur de vérité de Q . L'implication est à la base du raisonnement mathématique. En partant d'une assertion P (ou de plusieurs), une démonstration aboutit à un résultat Q . Si cette démonstration est faite sans erreur, alors $P \rightarrow Q$ est vraie et on notera $P \Rightarrow Q$ (ce qui signifie que si P est vraie, alors Q est vraie). Dans ce cas, on dit que P est une condition suffisante et Q une condition nécessaire. On peut remarquer que l'implication est transitive, c'est-à-dire que si P implique Q et Q implique R , alors P implique R .
- L'équivalence de P et Q , notée $P \leftrightarrow Q$, est l'assertion qui est vraie uniquement si $P \rightarrow Q$ et $Q \rightarrow P$ sont toutes deux vraies. Dans le cas où $P \leftrightarrow Q$ est vraie on dit que P est Q sont équivalentes et on note $P \Leftrightarrow Q$ (ce qui signifie que P et Q sont, soit toutes deux vraies, soit toutes deux fausses). Dans ce cas, on dit que Q est une condition nécessaire et suffisante de P .

On peut résumer ce qui précède, en utilisant la table de vérité suivante :

P	Q	\overline{P}	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

Deux assertions qui ont même table de vérité sont équivalentes en vertue de la définition 6.9.

7.2 Règles de calcul propositionnel

Avec le théorème qui suit, on résume quelques règles de calcul.

Theorem 7.2 *Soient P, Q, R des propositions. On a les équivalences :*

(i) *commutativité :*

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

$$(P \vee Q) \Leftrightarrow (Q \vee P)$$

(ii) *associativité*

$$(P \wedge (Q \wedge R)) \Leftrightarrow ((P \wedge Q) \wedge R)$$

$$(P \vee (Q \vee R)) \Leftrightarrow ((P \vee Q) \vee R)$$

(iii) *distributivité* :

$$(P \wedge (Q \vee R)) \Leftrightarrow ((P \wedge Q) \vee (P \wedge R))$$

$$(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$$

(iv) *négations* :

$$(\overline{\overline{P}}) \Leftrightarrow (P)$$

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

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

$$(P \rightarrow Q) \Leftrightarrow (\overline{Q} \rightarrow \overline{P})$$

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

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

Les équivalences $(\overline{P \wedge Q}) \Leftrightarrow (\overline{P} \vee \overline{Q})$ et $(\overline{P \vee Q}) \Leftrightarrow (\overline{P} \wedge \overline{Q})$ sont appelées lois de De Morgan.

7.3 Clauses et formes normales

Les clauses sont des expressions logiques de grand intérêt en informatique. On les définit ainsi.

Definition 7.3

- Une clause conjonctive est de la forme :

$$l_1 \wedge l_2 \wedge \cdots \wedge l_n,$$

- une clause disjonctive est de la forme :

$$l_1 \vee l_2 \vee \cdots \vee l_n,$$

où les l_i sont des littéraux, c'est-à-dire des atomes ou des négations d'atomes.

Par exemple, la proposition “tous les ascenseurs fonctionnent” peut être considérée comme une clause conjonctive, “tous les ascenseurs fonctionnent sauf le 5” aussi. Par contre, “deux ou trois ascenseurs ne fonctionnent pas” n'est pas une clause.

Note . Le plus souvent, le terme clause renvoie à la clause disjonctive.

Definition 7.4 *On appelle forme normale une conjonction de clauses disjonctives.*

Pour exemple d'une forme normale, nous avons

$$(A \vee \neg B \vee \neg C) \wedge (\neg D \vee E \vee F).$$

Par contre,

$$A \wedge (B \vee (C \wedge D))$$

n'est pas une forme normale car un "et" est imbriqué dans un "ou". On a le théorème suivant.

Theorem 7.5 *Toute formule admet une forme normale qui lui est équivalente.*

Son application se fera par l'utilisation des règles de calcul précédentes. Notre attention devrait d'ailleurs se porter très fortement sur l'équivalence

$$(P \rightarrow Q) \Leftrightarrow (\overline{P} \vee Q).$$

Si nous prenons une clause conjonctive et qu'elle implique un littéral q ,

$$l_1 \wedge \cdots \wedge l_n \Rightarrow q,$$

une formulation équivalente est

$$\overline{l_1 \wedge \cdots \wedge l_n} \vee q,$$

autrement dit,

$$\bar{l}_1 \vee \cdots \vee \bar{l}_n \vee q.$$

Nous obtenons alors une nouvelle clause dont un seul littéral est positif. Il s'agit d'une clause de Horn qu'on définit ainsi.

Definition 7.6 *Une clause de Horn est une clause disjonctive avec au plus un littoral positif (non négatif).*

Notons que cette définition n'exclue pas de n'avoir que des littéraux négatifs

$$\bar{l}_1 \vee \cdots \vee \bar{l}_n,$$

ce qui revient, par les lois de Morgan, à vérifier

$$l_1 \wedge \cdots \wedge l_n.$$

Ces considérations sont la base de la programmation en PROLOG.

7.4 Modus ponens – Principe de résolution de Robinson

La règle du modus ponens s'écrit

$$\frac{p \quad p \rightarrow q}{q},$$

et se lit : de p et de “ p implique q ”, je déduis q . Puisque l'implication $p \Rightarrow q$ est équivalente à “non p ou q ”, la règle du modus ponens s'écrit

$$\frac{p \quad (\neg p \vee q)}{q}.$$

On peut généraliser cette règle d'inférence logique aux clauses par le principe de résolution de Robinson qui suit. Partons d'une clause contenant p ,

$$(p \vee l_1 \vee \dots \vee l_n)$$

et d'une implication de p ,

$$(p \Rightarrow m_1 \vee \dots \vee m_k) \iff (\neg p \vee m_1 \vee \dots \vee m_k),$$

alors on a $l_1 \vee \dots \vee l_n$ ou si p , le modus ponens nous dit qu'alors on a $m_1 \vee \dots \vee m_k$. Ainsi, on obtient $(l_1 \vee \dots \vee l_n \vee m_1 \vee \dots \vee m_k)$. On peut résumer le principe de résolution de Robinson ainsi

$$\frac{(p \vee l_1 \vee \dots \vee l_n) \quad (\neg p \vee m_1 \vee \dots \vee m_k)}{(l_1 \vee \dots \vee l_n \vee m_1 \vee \dots \vee m_k)}.$$

8 Prédicats, quantificateurs

Nous nous contenterons d'une définition intuitive (ou naïve) de la notion d'ensemble.

Un ensemble est une collection d'objets possédant des propriétés communes, ces objets sont les éléments de l'ensemble.

Un peu plus précisément, la théorie des ensembles est une théorie portant sur des objets (appelés ensembles) dans laquelle figurent les signes $=$ et \in (ce qui se lit “ appartient à”) et qui vérifie certains axiomes.

8.1 Quantificateurs

La théorie des ensembles est une théorie “quantifiée” :

- Le quantificateur universel “quel que soit” ou “pour tout” noté \forall utilisé pour signifier que tout élément x d'un ensemble E vérifie une propriété $P(x)$, la syntaxe étant :

$$(\forall x \in E) (P(x)).$$

- Le quantificateur existentiel “il existe” noté \exists pour signifier qu'il existe au moins un élément x de E vérifiant la propriété $P(x)$, la syntaxe étant :

$$(\exists x \in E) | (P(x)).$$

- Pour signifier qu'il existe un et un seul x dans E vérifiant la propriété $P(x)$, on utilisera la notation :

$$(\exists!x \in E) \mid (P(x)).$$

On résume...

\in	appartient à	$x \in A$	x est dans A
\forall	pour tout, quelque-soit	$\forall x \in A$	pour tout x dans A
\exists	il existe, pour un certain	$\exists x \in A$	il existe x dans A

8.1.1 Ordre des quantificateurs

En utilisant les quantificateurs, il faudra faire attention à l'ordre d'apparition de ces derniers. Par exemple les assertions suivantes, où f est une fonction à valeurs réelles définie sur un ensemble E :

$$\forall x \in E, \exists M > 0 \mid f(x) < M$$

et

$$\exists M > 0 \mid \forall x \in E, f(x) < M.$$

ne sont pas équivalentes. La première assertion signifie que pour tout élément x de E il existe un réel $M > 0$ qui dépend à priori de x (il faudrait donc le noter $M(x)$) tel que $f(x) < M$ (par exemple $M(x) = f(x) + 1$ convient), alors que la seconde signifie qu'il existe un réel $M > 0$, indépendant de x dans E , tel que, pour tout x , $f(x) < M$, ce qui n'est pas la même chose. On pourra cependant noter la propriété suivante.

Proposition 8.1

$$(\exists a \ \forall b \mid P(a, b)) \quad \Rightarrow \quad (\forall b \ \exists a \mid P(a, b)).$$

Nous avons vu que la réciproque est fausse.

8.1.2 Négation

Il est intéressant de pouvoir formuler la négation d'une proposition. Soient $P(x)$ et $Q(x)$ des prédictats.

- La négation de l'assertion $(\forall x \in E) \mid (P(x))$ est :

$$(\exists x \in E) \mid (\overline{P(x)})$$

en utilisant le symbole \mid qui se lit “tel que” utilisé pour traduire le fait que x est tel que la propriété $\overline{P(x)}$ est vérifiée.

- La négation de $(\exists x \in E) \mid (P(x))$ est :

$$(\forall x \in E) \left(\overline{P(x)} \right).$$

Ainsi, pour nier un prédicat, par exemple l'existence d'un majorant M

$$\exists M > 0 \mid \forall x \in E, f(x) < M,$$

il suffit de nier chaque quantificateur, l'un après l'autre, et de nier la proposition qui suit :

$$\forall M > 0, \exists x \in E, |f(x)| \geq M,$$

Nous verrons qu'il n'est pas toujours facile de traduire la négation d'une assertion en utilisant les quantificateurs.

Par exemple pour traduire le fait qu'une suite $(u_n)_{n \in \mathbb{N}}$ de nombres réels est convergente vers un réel ℓ nous écrirons :

$$(\exists \ell \in \mathbb{R}) \mid (\forall \varepsilon > 0, \exists n_0 \in \mathbb{N} \mid \forall n \geq n_0, |u_n - \ell| < \varepsilon)$$

ce qui signifie qu'il existe un réel ℓ tel que quel que soit la précision $\varepsilon > 0$ que l'on choisisse l'écart entre u_n et ℓ (soit $|u_n - \ell|$) est inférieur à ε à partir d'un certain rang n_0 .

La négation de cette assertion s'écrit :

$$(\forall \ell \in \mathbb{R}), (\exists \varepsilon > 0, \forall n_0 \in \mathbb{N}, \exists n \geq n_0 \mid |u_n - \ell| \geq \varepsilon)$$

Pour résumer,

Quantificateurs	Négation	Stratégie
$\forall x, p(x)$	$\exists x, \bar{p}(x)$	Trouver un contre-exemple
$\exists x, p(x)$	$\forall x, \bar{p}(x)$	Tout x contredit $p(x)$
Connecteurs	Négation	Stratégie
$p(x)$ et $q(x)$	$\bar{p}(x)$ ou $\bar{q}(x)$	Contredire l'un des prédictats
$p(x)$ ou $q(x)$	$\bar{p}(x)$ et $\bar{q}(x)$	Contredire les deux prédictats
$p(x) \Rightarrow q(x)$	$p(x)$ et $\bar{q}(x)$	Trouver un élément vérifiant $p(x)$ et contredisant $q(x)$

8.1.3 Ensembles et logique : Diagrammes de Venn

Lorsqu'on évalue un prédicat, on peut toujours se ramener à la théorie des ensembles, en considérant l'ensemble des éléments qui satisfont le prédicat.

Dans l'exemple précédent, le prédicat $x + 6 = 10$ est vrai sur l'ensemble solution de l'équation $x + 6 = 10$, i.e. $x \in \{4\}$.

Formalisons plus sérieusement ces idées par la proposition suivante.

Proposition 8.2 Supposons qu'un prédicat $P(x)$ est vrai pour tout $x \in A$, et faux sinon. En posant 0 et 1 les valeurs logiques, P a les mêmes valeurs, en fonction de x , que la fonction indicatrice

$$\mathbb{I}_A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases}$$

Il y a donc une correspondance entre un prédicat et un ensemble.

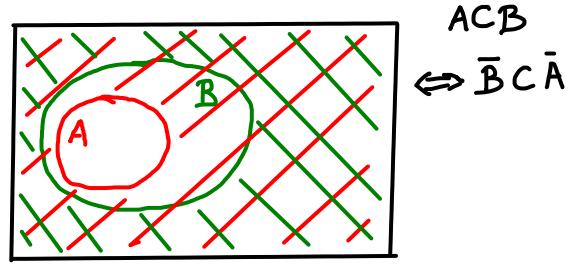
On donne alors le tableau des connecteurs logiques suivants.

Connecteur		Négation	Conjonction	Disjonction (non exclusif)	Implication	Équivalence
Notations	P (resp. Q)	Non P $\neg P$ \bar{P}	P et Q $P \wedge Q$	P ou Q $P \vee Q$	$P \Rightarrow Q$	$P \iff Q$
Diagramme	$x \in A, x \in B$ A (resp. B) est l'ensemble des x vérifiant P (resp. Q)	\bar{A}	$A \cap B$	$A \cup B$	$A \subset B$	$A = B$

On note que la proposition $P \Rightarrow Q$ ne donne aucune valeur de vérité à la proposition $Q \Rightarrow P$. Il est intéressant de garder en mémoire cette proposition.

Proposition 8.3 *L'implication $P \Rightarrow Q$ est équivalente à $\bar{Q} \Rightarrow \bar{P}$.*

Proof. Évident d'après le dessin.



8.1.4 Skolémisation

Nous avons vu qu'il était d'usage de commencer un prédicat par les quantifications. On appelle cette formulation une forme prénexe. L'enjeu de la Skolémisation est de réduire le nombre de variables dans un prédicat, en particulier les variables quantifiées par \exists . Par exemple, lorsque nous avons écrit la tautologie

$$\forall x \in E, \exists M > 0 \mid f(x) < M,$$

la quantité M dépend directement de $f(x)$ et on pourrait écrire, par exemple $M = f(x) + 1 = g(x)$. Ainsi, on peut écrire ce prédicat

$$\forall x \in E, f(x) < g(x).$$

Ainsi, pour Skolémiser une expression F , on procède par les étapes suivantes.

- On transforme F en une forme prénexe.
- On remplace toute variable quantifiée existentiellement par un symbole de fonction dont les arguments sont les variables quantifiées universellement qui précèdent notre variable.

- On supprime les quantificateurs existentiels qui sont devenus inutiles.

Par exemple, prenons le prédicat

$$(\forall x \ p(x)) \wedge (\exists y \ q(y)) \rightarrow \exists y \ (p(y) \wedge q(y)).$$

Écrivons la forme prénexe

$$\begin{aligned} & \neg[(\forall x \ p(x)) \wedge (\exists y \ q(y))] \vee [\exists z \ (p(z) \wedge q(z))] \\ \iff & [(\exists x \ \neg p(x)) \vee (\forall y \ \neg q(y))] \vee [\exists z \ (p(z) \wedge q(z))] \\ \iff & \exists x \forall y \exists z \ (\neg p(x) \vee \neg q(y)) \vee (p(z) \wedge q(z)). \end{aligned}$$

Skolémisons

$$\forall y \ (\neg p(a) \vee \neg q(y)) \vee (p(f(y)) \wedge q(f(y))),$$

où x devient une constante a bien choisie.

La skolémisation sera une étape fondamentale pour l'utilisation des classes de Horn et du principe de résolution en Prolog.

9 Algèbre de Boole

Une algèbre de Boole est la donnée d'un triplet $(\Omega, 0, 1)$ et de

- une opération unaire $b \in \Omega \mapsto \bar{b} \in \Omega$, parfois noté \bar{b} ,
- deux opérations binaires notées \bullet et $+$ de $\Omega \times \Omega$ dans Ω

vérifiant, pour tout $(a, b, c) \in \Omega$, les propriétés suivantes :

$$\begin{array}{ll} a \bullet (b \bullet c) = (a \bullet b) \bullet c & a + (b + c) = (a + b) + c \\ a \bullet b = b \bullet a & a + b = b + a \\ a \bullet (b + c) = (a \bullet b) + (a \bullet c) & a + (b \bullet c) = (a + b) \bullet (a + c) \\ a + 0 = a & a \bullet 1 = a \\ a \bullet (-a) = 0 & a + (-a) = 0 \end{array}$$

En utilisant ces axiomes, on voit que, pour tout $a \in \Omega$ on a $a + 0 = 0 + a = a$: on dit que 0 est élément neutre pour la loi $+$. On voit de même que $a \bullet 1 = 1 \bullet a = a$: 1 est neutre pour la loi \bullet .

9.1 Lien avec la logique mathématique

Definition 9.1 Soit B une algèbre de Boole. Une fonction booléenne à n variables (ou arguments) est une application de B^n dans B .

Souvent $B = \{0, 1\}$, une telle fonction est définie sur un ensemble à 2^n éléments, elle peut donc être définie à l'aide d'un tableau appelé table de vérité.

Exemple. La table de vérité de $+$ est

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

On reconnaît la table de vérité du *ou*.

La table de \bullet est

A	B	$A \bullet B$
0	0	0
0	1	0
1	0	0
1	1	1

c'est aussi la table de multiplication des entiers modulo 2. On reconnaît la table de vérité du *et*.

9.1.1 Loi de Morgan

On montre que

$$-(a \bullet b) = (-a) + (-b) \quad \text{et} \quad -(a + b) = (-a) \bullet (-b) .$$

Ces relations sont les lois de De Morgan.

9.2 Tableau de Karnaugh

Les tableaux de Karnaugh permettent de représenter facilement des expressions booléennes.

9.2.1 Cas de deux variables

Le tableau de Karnaugh comprend quatre cases, correspondant aux quatre produits $ab, a\bar{b}, \bar{a}b, \bar{a}\bar{b}$.

		b	\bar{b}
		a	\bar{a}
ab	$a\bar{b}$		
$\bar{a}b$	$\bar{a}\bar{b}$		

La première ligne correspond à $ab + \bar{a}b = a(b + \bar{b}) = a1 = a$, la deuxième ligne correspond à $\bar{a}b + \bar{a}\bar{b} = \bar{a}(b + \bar{b}) = \bar{a}1 = \bar{a}$. De même, la première colonne est b et la deuxième est \bar{b} . Pour représenter une expression dans un tableau de Karnaugh, on colore les cases concernées par l'expression.

Exemple.

L'expression $\bar{a} + b$ est représentée par le tableau suivant, dans lequel on colore la ligne \bar{a} et la colonne b .

	b	\bar{b}
a		
\bar{a}		

9.2.2 Cas de trois variables

Le tableau de Karnaugh comprend huit cases, correspondant aux huit produits : $abc, a\bar{b}c, ab\bar{c}, a\bar{b}\bar{c}, \bar{a}bc, \bar{a}\bar{b}c, \bar{a}b\bar{c}, \bar{a}\bar{b}\bar{c}$.

		b	\bar{b}		
		abc	$a\bar{b}c$	$ab\bar{c}$	$a\bar{b}\bar{c}$
		$\bar{a}bc$	$\bar{a}\bar{b}c$	$\bar{a}b\bar{c}$	$\bar{a}\bar{b}\bar{c}$
c	b				
	\bar{b}				
\bar{a}	bc				
	$b\bar{c}$				
a	$\bar{b}\bar{c}$				
	$\bar{b}c$				

Exemple.

- L'expression c est représentée par le tableau suivant :

		bc	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
		a			
		\bar{a}			
	bc				
	$b\bar{c}$				
	$\bar{b}\bar{c}$				
	$\bar{b}c$				

- Pour l'expression $\bar{a} + b$, on réunit la ligne \bar{a} et les colonnes b (+ équivaut à OU et à réunion).

		bc	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
		a			
		\bar{a}			
	bc				
	$b\bar{c}$				
	$\bar{b}\bar{c}$				
	$\bar{b}c$				

- Pour $\bar{a}c$, on prend l'intersection de la ligne \bar{a} avec les colonnes c (• équivaut à ET et à intersection).

		bc	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
		a			
		\bar{a}			
	bc				
	$b\bar{c}$				
	$\bar{b}\bar{c}$				
	$\bar{b}c$				

- Le tableau suivant

	bc	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$	
a	■				
\bar{a}	■■	■	■	■	■
	bc				\bar{a}

permet de retrouver l'expression $\bar{a} + bc$.

9.2.3 Cas de quatre variables

Le tableau de Karnaugh comprend alors seize cases, correspondant aux seize produits :

$$abcd, a\bar{b}cd, ab\bar{c}d, a\bar{b}\bar{c}d, \bar{a}bcd, \bar{a}\bar{b}cd, \bar{a}\bar{b}\bar{c}d, abc\bar{d}, a\bar{b}c\bar{d}, ab\bar{c}\bar{d}, a\bar{b}\bar{c}\bar{d}, \bar{a}bc\bar{d}, \bar{a}\bar{b}c\bar{d}, \bar{a}\bar{b}\bar{c}d, \bar{a}\bar{b}\bar{c}\bar{d}.$$

		c	cd	$c\bar{d}$	\bar{c}	$\bar{c}d$	$\bar{c}\bar{d}$
		d	d	d	d	d	d
a	ab						
a	$a\bar{b}$						
\bar{a}	$\bar{a}b$						
\bar{a}	$\bar{a}\bar{b}$						

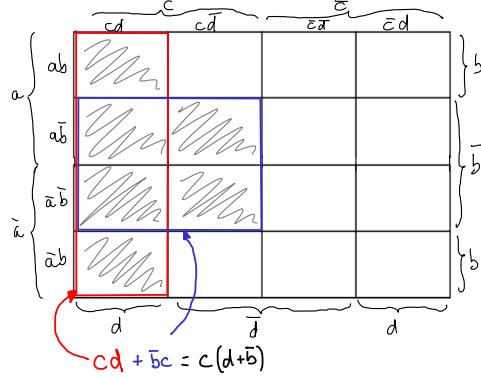
Prenons l'exemple de l'expression

$$abcd + \bar{a}bcd + \bar{b}cd + \bar{b}c\bar{d}.$$

de tableau de Karnaugh

		c	cd	$c\bar{d}$	\bar{c}	$\bar{c}d$	$\bar{c}\bar{d}$
		d	d	d	d	d	d
a	ab	W					
a	$a\bar{b}$	W	W				
\bar{a}	$\bar{a}b$	W	W	W			
\bar{a}	$\bar{a}\bar{b}$	W					

On obtient l'expression simplifiée $c(d + \bar{d})$:



10 Récurrence et récursivité

Induction et récursion sont deux concepts intimement liés, deux facettes d'une même idée. Dans l'usage courant, les deux mots sont employés de façon parfois interchangeable, et en tout cas avec une frontière assez floue.

Formellement, l'induction (la récurrence) est une technique de preuve mathématique employée pour démontrer des théorèmes sur les nombres naturels ou sur d'autres structures infinies munies d'un ordre bien fondé (souvent, des structures définies récursivement).

D'un autre côté, la récursion est une technique de définition et construction d'objets mathématiques (fonctions, ensembles, etc.).

10.1 Preuve par récurrence

Dans sa forme la plus simple, le principe d'induction est une technique de preuve qui permet de prouver qu'un prédicat $P(n)$ est vrai pour tous les entiers ($\forall n, P(n)$).

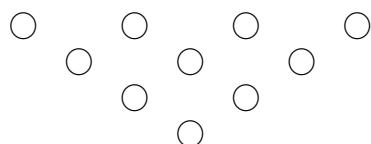
Une preuve par induction (on dit aussi preuve par récurrence) procède en deux étapes :

- On démontre que le prédicat est vrai pour un nombre fini de cas initiaux ;
- On démontre que si le prédicat est vrai pour un cas quelconque, alors il est vrai aussi pour le cas suivant

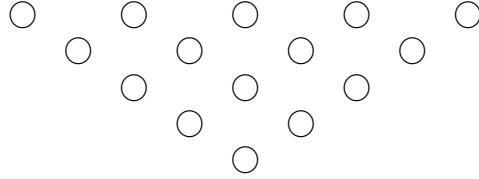
$$P(n) \Rightarrow P(n+1).$$

10.2 Mise en place d'un raisonnement récursif en mathématiques

Considérons le nombre de quilles au bowling



Posons $\mathcal{Q}(n)$ le nombre de quilles nécessaires pour faire n rangs. On a $\mathcal{Q}(1) = 1$, $\mathcal{Q}(2) = 3$, $\mathcal{Q}(3) = 6$, et $\mathcal{Q}(4) = 10$. Ajoutons une ligne de quilles :



On peut écrire

$$\mathcal{Q}(5) = 5 + \mathcal{Q}(4) = 15.$$

Supposons maintenant que l'on souhaite calculer $\mathcal{Q}(10)$. On posera l'ensemble des calculs jusqu'à une quantité connue :

$$\begin{aligned}\mathcal{Q}(10) &= 10 + \mathcal{Q}(9) \\ &= 10 + 9 + \mathcal{Q}(8) \\ &= 19 + 8 + \mathcal{Q}(7) \\ &= 27 + 7 + \mathcal{Q}(6) \\ &= 34 + 6 + \mathcal{Q}(5) \\ &= 40 + 15 = 55.\end{aligned}$$

10.3 Définitions mathématiques par récursion

De manière intuitive, une fonction (ou une propriété) récursive est une fonction qui est définie en termes d'elle-même. Ceci ne suffit pas à définir correctement une fonction, car par exemple l'égalité $f(x) = f(x)$ ne définit aucune fonction en particulier (toute fonction a cette propriété).

Prenons l'exemple de la factorielle d'un entier n . On pourra la définir de la façon suivante

$$n! = 1 \times 2 \times 3 \times \cdots \times n.$$

Problème : l'ellipse “...”. On peut préférer la définition récursive

$$\begin{cases} 1! = 1 \\ n! = n(n-1)! \end{cases}.$$

Cette définition, qui se réfère à elle-même, est dite récursive : la valeur de la fonction en n s'exprime au moyen de la valeur de la fonction en $n-1$.

10.4 Des preuves par récursion ?

Rares sont les preuves mathématiques présentées sous la forme de récursion, c'est-à-dire, où on va formellement prouver qu'un prédicat $P(n)$ est vrai au rang n car vrai à tous les rangs inférieurs.

Le raisonnement par récurrence lui est très clairement préféré car donnant à la propriété une véracité pour tout rang $n \in \mathbb{N}$, et non jusqu'à un certain rang k numérique prédéterminé.

On peut néanmoins réfléchir à un des théorèmes vu en arithmétique. Rappelons-le ici :

Theorem 10.1 Soit n un entier naturel. Si n n'est pas premier, il admet au moins un diviseur premier inférieur ou égal à \sqrt{n} .

Ainsi, ce théorème nous amène à tester la primalité d'un nombre n en vérifiant sa divisibilité par des nombres inférieurs à \sqrt{n} dont on peut tester la primalité.

10.5 Algorithme itératif

L'algorithme itératif repose sur l'utilisation des boucles pour itérer une action. Il y a principalement deux types de boucles : **for** et **while**.

10.5.1 Boucle for

On va répéter une action qui dépend de la valeur d'une variable **var** (appelée compteur) sachant qu'on veut qu'elle prenne valeur dans un intervalle d'entiers, un **itérable**, voire un fichier de donnée... Plein de choses sont itérables. Voici la syntaxe.

Pseudo-code	Résultat
Pour var de 1 à n : $< instructions >$	La variable var prend toutes les valeurs entières de 1 à n et pour chacune d'elles, exécution du bloc $< instructions >$
Pour var dans itérable : $< instructions >$	La variable var prend toutes les valeurs de itérable et pour chacune d'elles, exécution du bloc $< instructions >$

10.5.2 Boucle while

La boucle **while** permet d'itérer tant qu'une condition sur la variable compteur est remplie. Il n'y a pas d'itérable à priori. Il faut faire attention à ne pas lancer une boucle infinie. La variable compteur doit être indentée dans la boucle.

Pseudo-code	Résultat
Tant que condition : $< instructions >$	Tant que la condition condition est vraie, exécution du bloc $< instructions >$

10.5.3 La factorielle

Pour le calcul de la factorielle, l'algorithme itératif indique qu'il faut multiplier 1 par 2, stocker le résultat de ce calcul dans une variable, multiplier le contenu de cette variable par 3, ..., jusqu'à n .

Pour l'exemple, on pourra taper le code R suivant

```
> factoriel=function(n){
+ factorielle=1
+ for(i in 1:n){
+ factorielle=factorielle*(i) }
+ factorielle}
```

10.6 Algorithme récursif

Definition 10.2 *Un algorithme est récursif s'il s'appelle lui-même pour effectuer son traitement.*

Il est impératif de veiller à ce qu'un algorithme récursif se termine sous peine d'engendrer un processus infini.

L'algorithme récursif du calcul de la factorielle d'un nombre suit directement la définition de cette factorielle.

Pour l'exemple, on pourra taper le code R suivant

```
factorielr=function(n){  
  if(n==1){1}  
  else{n*factorielr(n-1)}  
}
```

10.7 Algorithme itératifs Vs récursif

Tout problème dont la résolution s'appuie sur un algorithme récursif peut aussi être résolu à l'aide d'un algorithme itératif (plus ou moins simplement), et vice versa.

Certains langages informatiques appelés langages fonctionnels, ne possèdent ni instruction d'affectation ni structure itérative.

Les langages qui privilégient les structures itératives (et l'affectation) sont dits impératifs. Python, R sont des langages multi-paradigme : ils permettent de programmer de faon impérative, objet ou fonctionnelle (les appels récursifs, par exemple, ne sont pas aussi bien optimisés en Python que dans les langages fonctionnels, et à contrario, en R, les boucles peuvent ne pas aboutir, cf TP d'analyse 3).

Part III

Théorie des graphes

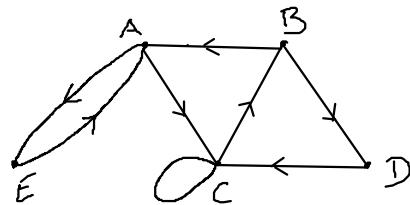
11 Introduction

Voici un exemple introductif pour imager les concepts.

Un site internet est composé de cinq pages notées A, B, C, D et E. En un clic, on peut passer d'une page à certaines autres selon les possibilités suivantes.

- (i) De la page A, on peut passer en un clic aux pages C et E.
- (ii) De la page B, on peut passer aux pages A et D.
- (iii) Depuis la page C, on peut accéder à la page B ou rester sur C.
- (iv) Quand on est sur la page D, on peut seulement aller sur la page C et de la page E, on ne peut aller que sur la page A.

On peut modéliser cette situation avec un graphique sagittal, constitué de l'ensemble $S=\{A;B;C;D;E\}$ des sommets, et de flèches orientées appelées arcs.



On peut représenter chaque flèche par un couple (X,Y) où X est le point de départ de la flèche et Y son point d'arrivée. On obtient ainsi une liste de couples, qui est une partie du produit cartésien $S^2 : (A,C), (A,E), (B,A), (B,D), (C,B), (C,C), (D,C), (E,A).$

On peut aussi faire un tableau donnant, pour chaque page, les pages qui peuvent être atteintes par un seul clic. On fait alors le tableau des successeurs à gauche. Une autre possibilité est de faire le tableau des prédécesseurs à droite, c'est-à-dire un tableau donnant, pour chaque page, les pages qui ont permis d'y arriver avec un seul clic.

Page (sommet)	Successseurs	Page (sommet)	Prédécesseurs
A	C, E	A	B, E
B	A, D	B	C
C	B, C	C	A, C, D
D	C	D	B
E	A	E	A

On peut aussi faire un tableau à double entrée en codant 1 s'il existe une possibilité en un seul clic, d' aller d'une page de départ (origine) à une page d'arrivée (extrémité), et en codant 0 s'il n'y a pas possibilité de passage.

		Extrémité				
		A	B	C	D	E
Origine	A	0	0	1	0	1
	B	1	0	0	1	0
	C	0	1	1	0	0
	D	0	0	1	0	0
	E	1	0	0	0	0

De façon évidente, ce tableau peut être associé à la matrice carrée d'ordre 5 suivante, appelée matrice d'adjacence.

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

12 Matrice d'adjacence

12.1 Définitions

Un graphe simple orienté est défini par un ensemble fini $S = \{a_1; a_2; \dots; a_n\}$ dont les éléments sont appelés sommets, et par un ensemble Γ de couples appartenant au produit cartésien S^2 dont les éléments sont appelés arcs.

Un couple $(a_i; a_j)$ appartient à Γ si et seulement si, sur le diagramme sagittal, on peut relier le sommet a_i au sommet a_j par une flèche orientée.

Le sommet a_i de l'arc est appelé origine et le sommet a_j est appelé extrémité. Si l'origine et l'extrémité sont confondues, l'arc est une boucle.

Soit $(a_i; a_j)$ un arc d'un graphe G :

- Le sommet a_j est un successeur de a_i et a_i est un prédécesseur de a_j .
- L'ensemble des successeurs d'un sommet a_i se note $\Gamma^+(a_i)$.
- L'ensemble des prédécesseurs d'un sommet a_j se note $\Gamma^-(a_j)$.

Reprenons l'exemple étudié en début de chapitre :

- Les successeurs de A sont C et E donc $\Gamma^+(A) = \{C;E\}$.
- De même, $\Gamma^+(B) = \{A;D\}$.
- Les prédécesseurs de A sont B et E donc $\Gamma^-(A) = \{B;E\}$.
- De même, $\Gamma^-(B) = C$.

Definition 12.1 (Matrice d'adjacence d'un graphe) Soit G un graphe orienté ayant n sommets

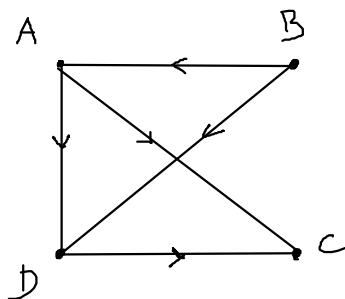
a_1, a_2, \dots, a_n . La matrice d'adjacence du graphe est la matrice carrée $M = (m_{i,j})$ d'ordre n telle que $m_{i,j} = 1$ si et seulement si $(a_i; a_j)$ est un arc de G , et $m_{i,j} = 0$ sinon.

Autre exemple

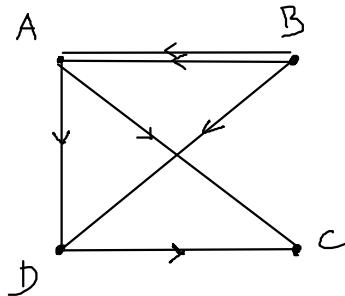
Si on considère la matrice d'adjacence

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

on a le graphe suivant



Note . Supposons que l'on ajoute un autre arc de B à A , comme une deuxième voie à une route. On aurait le graphe suivant



de matrice d'adjacence

$$M = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

12.2 Chemin et longueur dans un graphe orienté

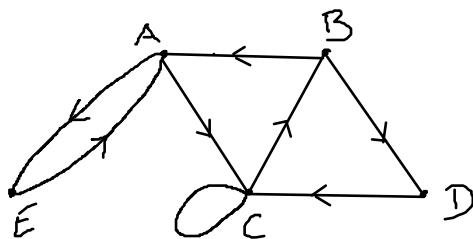
Definition 12.2 un chemin est une suite ordonnée de sommets dans laquelle chaque sommet, à part le premier, est un successeur du sommet qui le précède.

Un chemin hamiltonien est un chemin qui passe une seule fois par tous les sommets du graphe.

Un circuit est un chemin dans lequel le premier et le dernier sommet sont identiques.

La longueur d'un chemin est égale au nombre d'arcs qui constituent ce chemin. La longueur d'un chemin ayant n sommets est $n - 1$.

Rappelons le graphe de l'exemple initial.



En suivant des arcs, on peut aller du sommet E au sommet B en passant par les sommets A puis C. La suite ordonnée (E,A,C,B) est un chemin constitué de trois arcs. Ce chemin est de longueur 3. On peut remarquer que chaque arc de ce chemin peut être considéré comme un chemin de longueur 1.

(A,B,D) n'est pas un chemin car il n'y a pas d'arc qui permet de passer de A à B . (D,C,B,D) est un chemin dont le premier sommet est le même que le dernier sommet : c'est un circuit. (E,A,C,B,D) est un chemin qui passe par tous les sommets, une fois et une seule : on dit que c'est un chemin hamiltonien.

12.3 Nombre de chemins de longueur donnée

Proposition 12.3 Soit M la matrice d'adjacence d'un graphe orienté à n sommets a_1, a_2, \dots, a_n . Soit p un entier et $M^p = (m_{ij})$ la puissance p de la matrice M . Alors $m_{i,j}$ est le nombre de chemins de longueur p allant du sommet a_i au sommet a_j .

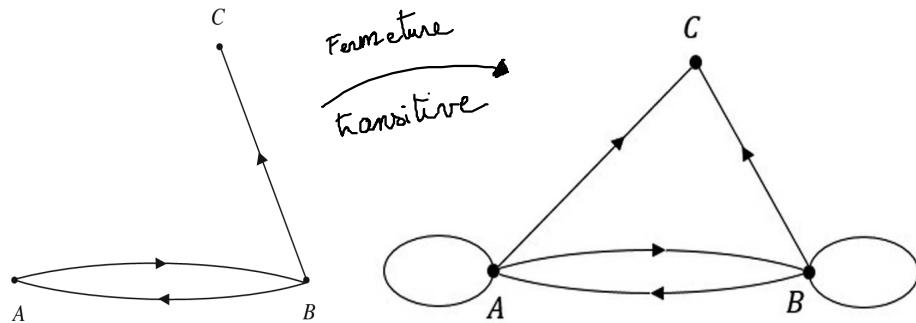
Dans notre exemple de site internet,

$$M = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad M^2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Il y a donc un chemin de longueur 2 qui va de A à B et deux chemins de longueur 2 de B à C .

12.4 Fermeture transitive

Faire la fermeture transitive d'un graphe consiste à rajouter tous les arcs (a_i, a_j) dès qu'il existe un chemin allant du sommet a_i au sommet a_j .



Il peut être difficile de faire la fermeture transitive d'un graphe sans oublier d'arcs. Mais en utilisant la matrice d'adjacence d'un graphe, et au prix de quelques calculs matriciels, on peut déterminer la matrice d'adjacence de la fermeture transitive. Il est alors plus facile de ne pas oublier d'arcs. Nous admettrons la propriété suivante.

Proposition 12.4 Soit M la matrice d'adjacence d'un graphe à n sommets et soit \tilde{M} la matrice d'adjacence de la fermeture transitive du graphe. Alors $\tilde{M} = M \oplus M^{[2]} \oplus \dots \oplus M^{[n]}$ (ou \oplus est l'addition booléenne, et $M^{[n]}$ est la puissance booléenne, on rappellera la définition dans ce qui suit).

L'addition et la multiplication (donc la puissance) booléenne revient à considérer les coefficients des matrices comme des booléens : on garde les zéros et on ramène à 1 tous les coefficients supérieurs. On comprendra avec l'exemple du graphe précédent dont la matrice d'adjacence est la suivante.

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

Le graphe possède $n = 3$ sommets donc d'après la propriété précédente, la matrice d'adjacence de la fermeture transitive est $\hat{M} = M \oplus M^{[2]} + M^{[3]}$. Les calculs donnent les résultats suivants. On a

$$M^{[2]} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

et

$$M^{[3]} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Il vient alors que

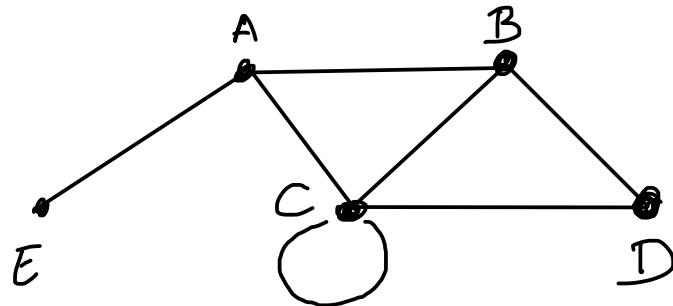
$$\begin{aligned} \hat{M} &= M \oplus M^{[2]} + M^{[3]} \\ &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

Connaissant maintenant la matrice d'adjacence de la fermeture transitive, il est facile de compléter le graphe.

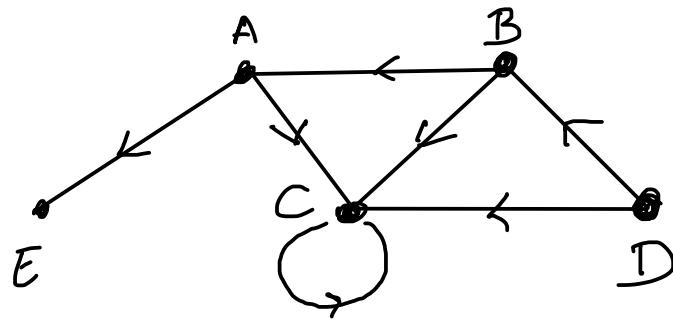
13 Algorithme de Dijkstra

13.1 Graphe non orienté

On peut supposer que tous les arcs sont “à double sens”. À ce moment là, plus besoin de flèche pour indiquer un successeur.



Ainsi, si A est successeur de B , A est aussi prédecesseur de B . Pour donner la matrice d'adjacence de ce graphe non orienté, on va donner la matrice A du graphe orienté arbitrairement



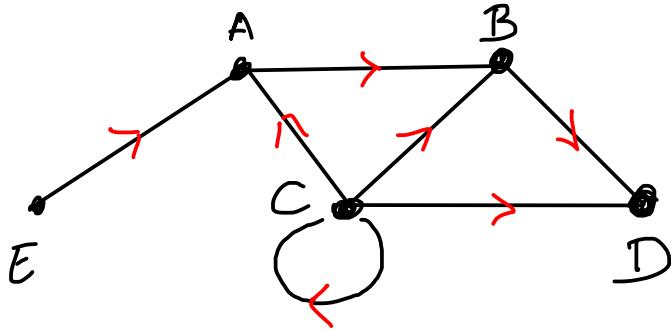
dont la matrice d'adjacence est

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

Notons que la transposée de A

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

est la matrice d'ajacence du graphe où tous les arcs sont orientés de manière opposée



La matrice d'adjacence du graphe non orienté sera la matrice symétrique

$$M = A + {}^t A.$$

Ici,

$$= \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

On notera qu'on a été obligé d'orienter deux fois la boucle sur C , d'où le coefficient 2 sur la diagonale.

Definition 13.1 *Un graphe non orienté est dit connexe si quels que soient les sommets U et V , il existe une chaîne (chemin d'un graphe non orienté) reliant U à V .*

13.2 Arbre

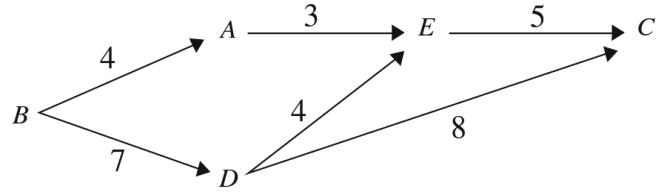
Un arbre est un graphe non orienté qui vérifie les propriétés suivantes.

- Connexité : il est toujours possible d'aller d'un sommet à l'autre par un chemin d'arêtes.
- Acyclique : il est impossible de partir d'un sommet et d'y revenir sans rebrousser chemin à un moment.

13.3 Graphe pondéré (ou valué)

Sur certains graphes orientés ou non, il peut être nécessaire d'attribuer une valeur à chacun des arcs : on obtient alors un graphe pondéré ou valué. La valeur d'un chemin est la somme des valeurs des arcs qui constituent le chemin. Il est alors possible de chercher le chemin de valeur minimale ou maximale reliant un sommet à un autre.

Exemple. Considérons le graphe suivant que l'on pondère arbitrairement pour cet exemple :



pour relier B à C, il y a trois chemins possibles.

- Le chemin (B,A,E,C) de valeur $4 + 3 + 5 = 12$.
- Le chemin (B,D,E,C) de valeur $7 + 4 + 5 = 16$.
- Le chemin (B,D,C) de valeur $7 + 8 = 15$.

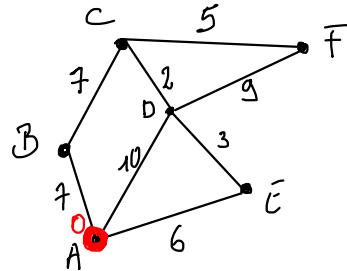
Le chemin minimal est (B,A,E,C), le chemin maximal est (B,D,E,C).

13.4 Recherche de chemin le plus court – Algorithme de Dijkstra

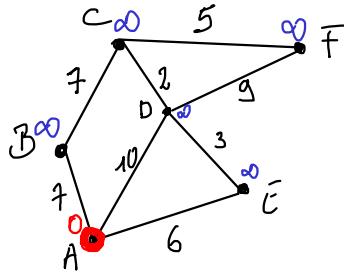
Sur un graphe pondéré, orienté ou non, on va s'intéresser aux chemins les plus depuis une source vers tous les sommets connexes. Cet algorithme permettra, par exemple, de connaître le chemin le plus court, en temps ou en distance..., d'une ville à une autre, d'une station de train à une autre...

13.4.1 l'Algorithme

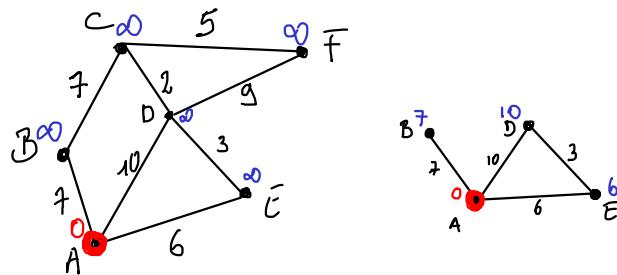
Sur un graphe pondéré, orienté ou non, on se donne un sommet d'entrée (sommet source), disons A . Sa distance à lui même est nulle.



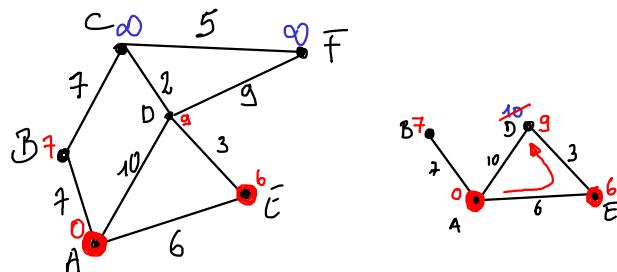
Le problème est de donner le chemin le plus court vers tous les sommets du graphe. On suppose au départ que tous les sommets sont à une distance infinie de la source.



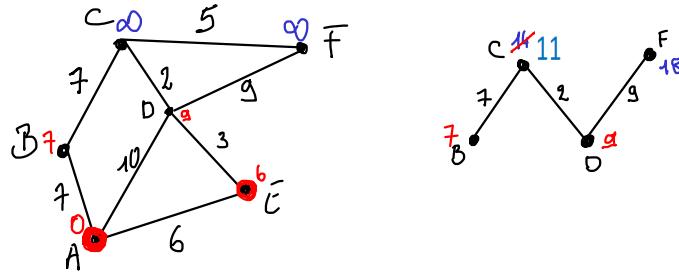
On donne la valeur provisoire de l'arc à chaque successeur direct de A .



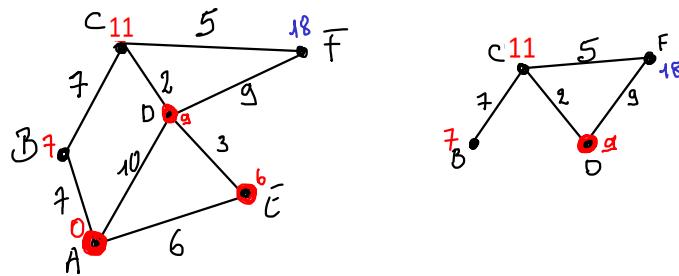
On regarde si le passage par d'autres arcs amèneraient à une longueur inférieure. Si tel est le cas, on remplace la distance précédente par celle-ci et on rougit ce chemin. Par exemple, (A, E, D) est plus court que (A, D) , D est donc à une longueur 9, on rougit E . À contrario, (A, E) est plus court que (A, D, E) , et, de manière évidente, le chemin le plus court pour aller à B est (A, B) . On a nos premières longueurs.



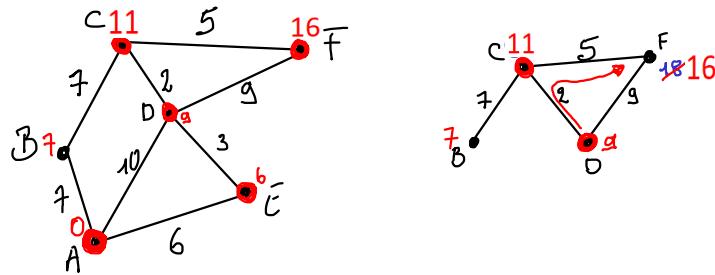
Considérons simultanément C et F , les suivants de B et D . C est moins loin de A en passant par D que par B . Sa longueur provisoire est 11, celle de F , à ce stade des 18.



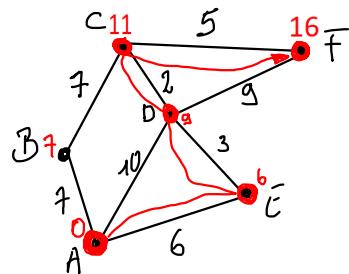
En ce qui concerne C , on doit passer par D pour avoir le chemin le plus court. Il en est de même pour F . D est rougi, on a la distance minimale de C .



Enfin, on raccourcit la distance de F en passant par C . On rougit C .



Il reste à sauter de point rouge en point rouge pour trouver le chemin le plus court de A vers F : (A, E, D, C, F) .



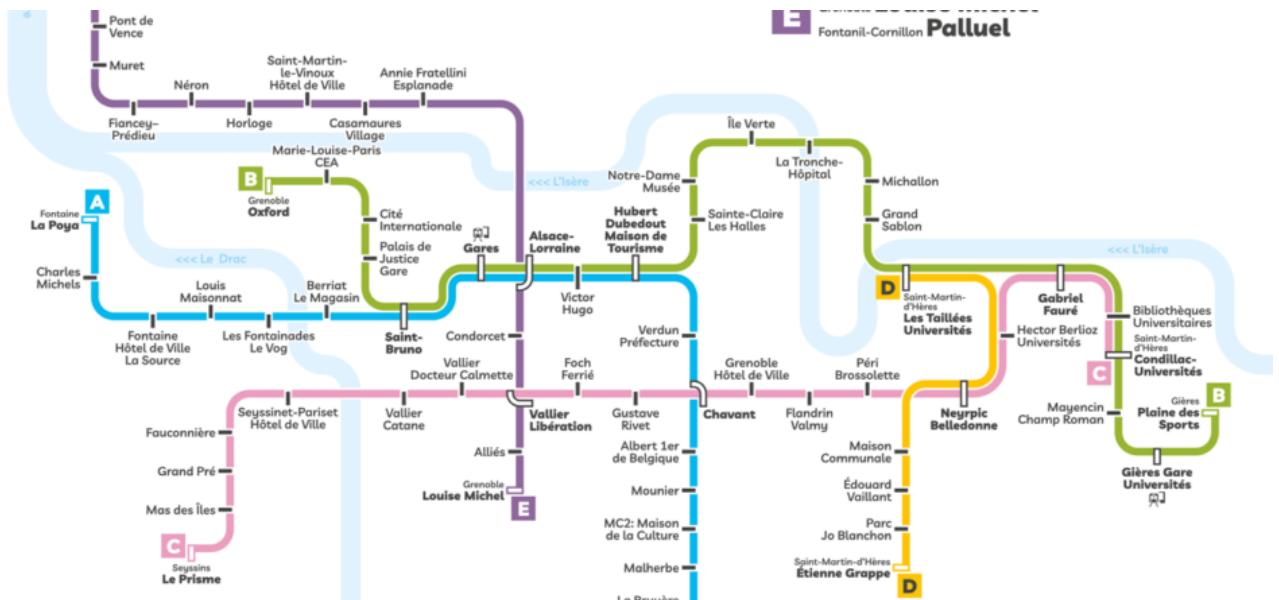
Notons que naturellement, le chemin le plus court de A à C est (A, E, D, C) . Nous avons également la note suivante.

Note. Un axiome courant en mathématiques dit que si on a le chemin le plus court entre un point et un autre, alors, si on prend un point de ce chemin, la suite du chemin est encore le chemin le plus court de ce point vers l'extrémité.

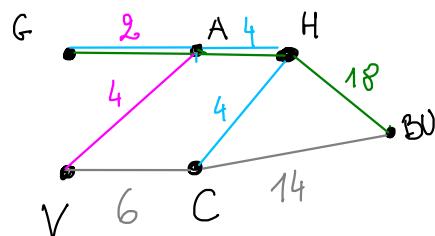
Ici, cela signifie, par exemple, de D à F , le chemin le plus court sera (D, C, F) .

13.4.2 Exemple : le tram de Grenoble

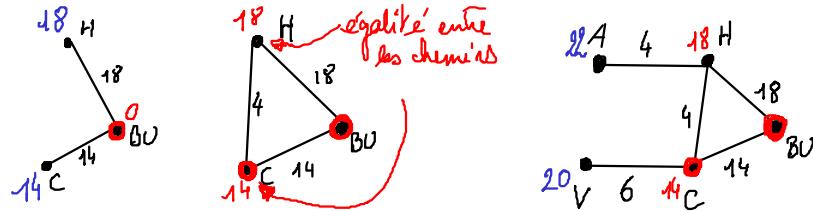
On voudrait connaître le chemin le plus court pour aller d'un point à un autre par le tram de Grenoble.



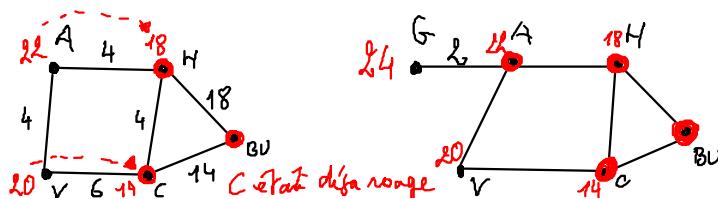
On va considérer que le temps de parcours entre deux stations est 2 minutes, qu'on est en heure de pointe et qu'un changement ne fait pas perdre de temps. On résume le plan ci-dessus sur les noeuds des lignes, on oublie le D qui est inefficace et on souhaite partir de Bibliothèques Universitaires. On a donc le graphe non orienté pondéré par le temps de parcours



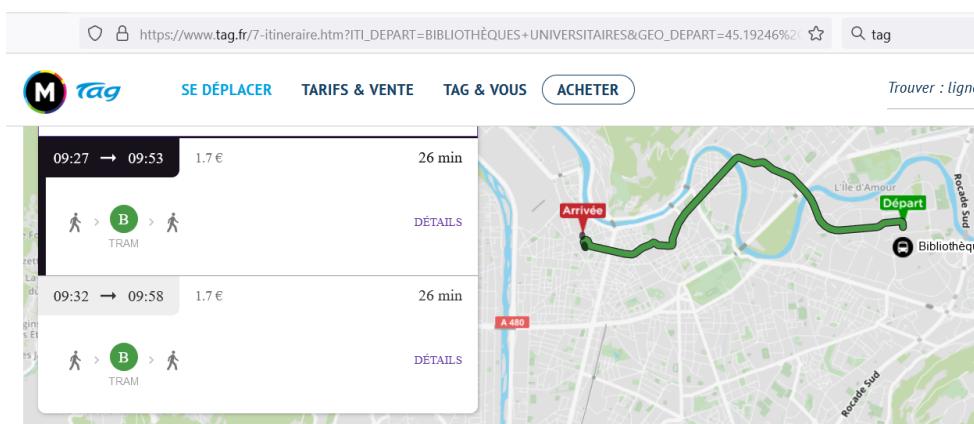
On souhaite aller à la gare. Assurons-nous, par l'algorithme de Dijkstra que le B est bien le plus court.



Il suit que



Il reste à sauter de pastille rouge en pastille rouge (tant qu'on garde un chemin minimal). On peut donc prendre le B pour l'ensemble du trajet, ce qui évite le changement et un risque d'attente, bien que le C puis le A soit une proposition équivalente si on considère qu'il n'y a pas de temps d'attente.
Il est à noter que nous sommes en accord avec le site TAG :



14 Dessin d'un graphe par niveaux

On appelle sommet de niveau 0 tout sommet qui n'a pas de prédécesseur. Si on note S_0 l'ensemble des sommets de niveau 0, alors on appelle sommet de niveau 1 tout sommet qui n'a pas de prédécesseur dans $S - S_0$ (ensemble S privé des éléments de S_0). On définit ensuite les sommets de niveau 2 et ainsi de suite...

Dans notre exemple initial de matrice d'adjacence

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

cherchons les sommets de niveau 0.

sommet	A	B	C	D	E
prédécesseur	B	\emptyset	A,E	B	A,D

B est le seul sommet qui n'a pas de prédécesseur, il est donc le seul sommet de niveau 0.

On a alors $S_0 = \{B\}$. On va maintenant chercher les sommets de niveau 1 : ce sont les sommets qui n'ont pas de prédécesseurs dans $S - S_0 = \{A; C; D; E\}$.

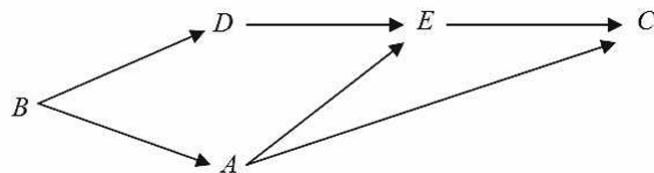
sommet	A	C	D	E
prédécesseur	\emptyset	A,E	\emptyset	A,D

A et D n'ont pas de prédécesseurs dans $S - S_0$, ils sont donc de niveau 1.

Il vient que $S_1 = \{A, D\}$. Les sommets de niveau 2 sont ceux qui n'ont pas de prédécesseurs dans $(S - S_0) - S_1 = \{C, E\}$. E n'a plus de prédécesseur alors que C a le sommet E comme prédécesseur. Donc E est de niveau 2, et enfin C est de niveau 3. Les niveaux des sommets sont résumés dans le tableau suivant.

Niveau	0	1	2	3
sommet	B	A,D	E	C

On peut alors faire le graphe en alignant verticalement les sommets de même niveau.



Attention, il ne s'agit plus du graphique sagittal, on a perdu des informations comme la flèche retour de E à A ou la boucle sur C.

15 Ordonnancement

La réalisation d'un projet passe par l'exécution de différentes tâches, de durées souvent différentes. Si certaines tâches peuvent être réalisées simultanément, d'autres nécessitent que certaines tâches aient

été réalisées antérieurement. Faire l'ordonnancement d'un projet consiste à organiser ce projet en respectant les contraintes d'antériorité des tâches tout en minimisant la durée totale de réalisation.

La méthode MPM (Méthode des potentiels metra) permet l'ordonnancement de projets, c'est la méthode que nous exposerons dans ce cours. Nous aurions pu choisir la méthode PERT, mais elle est plus complexe à mettre en oeuvre.

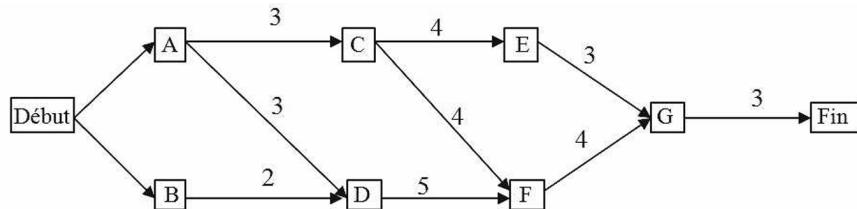
Exemple. Pour illustrer la méthode MPM, nous allons réaliser l'ordonnancement d'un projet fictif, comprenant des tâches que nous noterons A, B, C, D, E, F, G. Les contraintes d'antériorité et les durées de ces tâches sont consignées dans le tableau suivant.

Tâche	Durée (en jours)	Tâches antérieures
A	3	aucune
B	2	aucune
C	4	A
D	5	A, B
E	3	C
F	4	C, D
G	3	E, F

La première chose à faire est de définir le niveau de chaque tâche. Il est simple de voir que A et B sont de niveau 0, C et D de niveau 1, E et F de niveau 2 et G de niveau 3.

Niveau	0	1	2	3
sommet	A,B	C,D	E,F	G

Le graphe ordonné est le suivant



On peut remarquer que l'on a rajouté deux tâches fictives : Début et Fin ; et que l'on a pondéré le graphe en ajoutant les durées de chaque tâche.

Il est plutôt simple de calculer la durée minimale du projet (si vous ne la trouvez pas, nous la calculerons un peu plus loin). On peut se poser quelques questions.

- À quel moment peut-on commencer une tâche ?
- Est-ce qu'on peut retarder le moment de démarrer certaines tâches sans que cela ait d'impact sur la durée minimale du projet ?

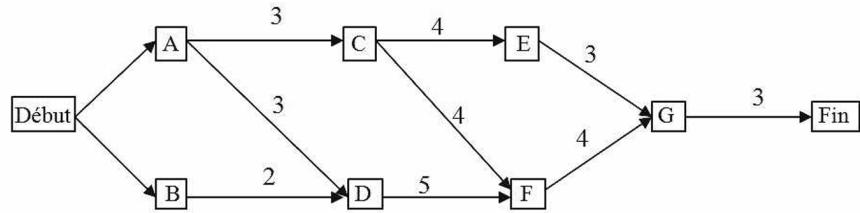
Pour répondre à ces questions, nous allons définir, pour chaque tâche, les notions de date au plus tôt et de date au plus tard.

15.1 Date au plus tôt

La date au plus tôt d'une tâche est la date minimale à laquelle on peut commencer la tâche, car toutes les tâches antérieures sont terminées.

Notation. Ces notions utilisent les durées des tâches. Pour la tâche x_j , nous la noterons $d(x_j)$. Nous noterons $t(x_j)$ la date au plus tôt d'une tâche x_j . $t(x_j)$ est le plus grand des nombres $t(x_i) + d(x_i)$ où x_i est une des tâches qui précèdent immédiatement la tâche x_j .

Reprenons notre dernier exemple.



Calculons les dates au plus tôt des tâches du projet :

- $t(A) = t(B) = 0$
- $t(C) = t(A) + d(A) = 0 + 3 = 3$
- D a deux prédécesseurs : A et B. On calcule
 - (i) $t(A) + d(A) = 0 + 3 = 3$,
 - (ii) $t(B) + d(B) = 0 + 2 = 2$,

$t(D)$ est le plus grand de ces deux nombres donc $t(D) = 3$.

- $t(E) = t(C) + d(C) = 3 + 4 = 7$
- F a deux prédécesseurs : C et D. On calcule
 - (i) $t(C) + d(C) = 7$,
 - (ii) $t(D) + d(D) = 3 + 5 = 8$,

Le plus grand de ces deux nombres est 8 donc $t(F) = 8$.

- Pour G, on calcule
 - (i) $t(E) + d(E) = 7 + 3 = 10$,
 - (ii) $t(F) + d(F) = 8 + 4 = 12$,
- et donc $t(G) = 12$.
- Pour finir, $t(Fin) = t(G) + d(G) = 12 + 3 = 15$.

On a ainsi calculé la date au plus tôt de la fin du projet qui est de 15 jours.

Note. Cette date au plus tôt de fin de projet ne correspond pas au chemin le plus court par l'algorithme de Dijkstra. En effet, dans un chemin le plus court, on ne se soucie pas d'avoir laissé le temps à chaque tâche (étape) d'avoir été faite.

15.2 Date au plus tard

La date au plus tard d'une tâche est la date maximale à laquelle on peut commencer la tâche sans que cela ne repousse la date de fin du projet.

Notation. Nous noterons $T(x_j)$ la date au plus tard d'une tâche x_j . $T(x_j)$ est le plus petit des nombres $T(x_k) - d(x_j)$ où x_k est une des tâches qui suit immédiatement la tâche x_j .

Pour poursuivre notre exemple, calculons les dates au plus tard des tâches du projet : on doit commencer par la fin puisqu'à chaque fois, on doit considérer les successeurs.

- On a bien sûr $T(Fin) = t(Fin) = 15$.
- $T(G) = T(Fin) - d(G) = 15 - 3 = 12$.
- $T(F) = T(G) - d(F) = 12 - 4 = 8$.
- $T(E) = T(G) - d(E) = 12 - 3 = 9$.
- $T(D) = T(F) - d(D) = 8 - 5 = 3$.
- C a deux successeurs : E et F. On calcule
 - (i) $T(E) - d(C) = 9 - 4 = 5$,
 - (ii) $T(F) - d(C) = 8 - 4 = 4$.

Le plus petit de ces deux nombres est 4, donc $T(C) = 4$.

- $T(B) = T(D) - d(B) = 3 - 2 = 1$.
- A a deux successeurs : C et D. On calcule
 - (i) $T(C) - d(A) = 4 - 3 = 1$,
 - (ii) $T(D) - d(A) = 3 - 3 = 0$.

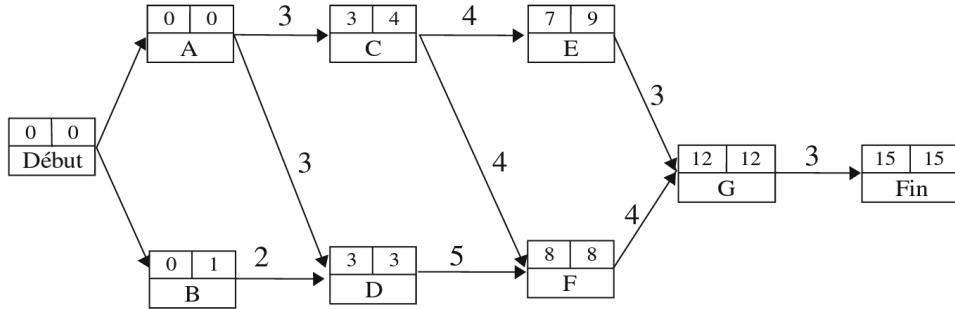
Donc $T(A) = 0$.

Pour chaque tâche, on va rajouter les dates au plus tôt et au plus tard en présentant chaque tâche par un rectangle.

$t(x_j)$	$T(x_j)$
x_j	

Par exemple, pour la tâche E :

7	9
E	



Examinons la tâche C : on a $t(C) = 3$ et $T(C) = 4$. Au plus tôt, on peut commencer à 3 jours et au plus tard à 4 jours. Pour cette tâche, on dispose donc d'une liberté d'un jour. Ce n'est pas ainsi pour d'autres tâches (comme D par exemple) pour lesquelles les dates au plus tôt et au plus tard sont les mêmes. Ces tâches sont qualifiées de tâches critiques. Et l'enchaînement A-D-F-G-Fin, qui ne contient que des tâches critiques, est qualifié de chemin critique.

Une tâche critique est une tâche dont les dates au plus tôt et au plus tard sont égales. Donc x_j est critique si et seulement si $t(x_j) = T(x_j)$. Un chemin critique est un chemin reliant le début à la fin et qui n'est constitué que de tâches critiques.

15.3 Marge totale d'une tâche

Definition 15.1 La marge totale d'une tâche c'est le retard maximum que l'on peut accepter sur la date de début de la tâche sans que cela ne retarde la date de fin du projet.

La marge totale d'une tâche x_j se note $MT(x_j)$ et elle vaut $MT(x_j) = T(x_j) - t(x_j)$.

Dans notre exemple, la tâche C n'est pas critique. On peut constater qu'on peut retarder le début de C d'un jour sans retarder la date de fin du projet (et c'est le maximum). En effet, si C commence à la date 4, cela n'empêcherait pas F de commencer à la date 8, E pourrait commencer à la date 8 et n'empêcherait pas G de commencer à la date 12. Ce retard maximum s'appelle la marge totale de la tâche.

La marge totale d'une tâche critique est nulle.