Arithmétique : en route pour la cryptographie
Un MOOC

Arithmétique : en route pour la cryptographie

Vous voulez comprendre l'arithmétique ? Vous souhaitez découvrir une application des mathématiques à la vie quotidienne ? Ce cours est fait pour vous ! De niveau première année d'université, vous apprendrez les bases de l'arithmétique (division euclidienne, théorème de Bézout, nombres premiers, congruences).

Vous vous êtes déjà demandé comment sont sécurisées les transactions sur Internet ? Vous découvrirez les bases de la cryptographie, en commençant par les codes les plus simples pour aboutir au code RSA. Le code RSA est le code utilisé pour crypter les communications sur internet. Il est basé sur de l'arithmétique assez simple que l'on comprendra en détails. Vous pourrez en plus mettre en pratique vos connaissances par l'apprentissage de notions sur le langage de programmation Python.

Comment utiliser ce support de cours ? Le but est de comprendre en détails les deux premiers chapitres ! Les autres chapitres sont là pour vous aider si vous en avez besoin.

- Chapitre 1. **Arithmétique.** Le cœur de nos préoccupations, nous étudierons tout pas à pas et en détails .
- Chapitre 2. **Cryptographie.** Notre motivation : comprendre le chiffrement RSA.
- Chapitre 3. **Algorithme.** Nous aurons besoin d'un petit peu de programmation pour «casser» des codes secrets. Nous n'étudierons pas ce chapitre en entier.
- Chapitre 4 et 5. Pour ceux qui ont besoin d'une petite remise à niveau, des rappels sur la **logique** et les **ensembles**.
- Des **exercices** pour l'arithmétique que l'on travaillera en profondeur. Et aussi pour ceux qui le souhaitent des exercices de remise à niveau pour la logique et les ensembles.

Plan de travail

- **Semaine 0**
  Remise à niveau sur les chapitres «Logique» et «Ensembles».
- **Semaine 1**
  Cryptographie : Le chiffrement de César
  Mathématiques : Division euclidienne
- **Semaine 2**
  Cryptographie : Le chiffre de Vigenère
  Mathématiques : pgcd
- **Semaine 3**
  Cryptographie : La machine Enigma et les clés secrètes
  Mathématiques : Théorème de Bézout
- **Semaine 4**
  Cryptographie : La cryptographie à clé publique
  Mathématiques : Nombres premiers
- **Semaine 5**
  Cryptographie : L'arithmétique pour RSA
  Mathématiques : Congruences
- **Semaine 6**
  Cryptographie : Le chiffrement RSA

Bon courage !
Arnaud Bodin & François Recher

Première partie

Le cours du MOOC

Arithmétique

Vidéo ■ partie 1. Division euclidienne et pgcd
Vidéo ■ partie 2. Théorème de Bézout
Vidéo ■ partie 3. Nombres premiers
Vidéo ■ partie 4. Congruences
Fiche d'exercices ♦ Arithmétique dans $\mathbb{Z}$

Préambule

Une motivation : l'arithmétique est au cœur du cryptage des communication. Pour crypter un message on commence par le transformer en un –ou plusieurs– nombres. Le processus de codage et décodage fait appel à plusieurs notions de ce chapitre :

- On choisit deux ***nombres premiers*** p et q que l'on garde secrets et on pose $n = p \times q$. Le principe étant que même connaissant n il est très difficile de retrouver p et q (qui sont des nombres ayant des centaines de chiffres).
- La clé secrète et la clé publique se calculent à l'aide de l'***algorithme d'Euclide*** et des ***coefficients de Bézout***.
- Les calculs de cryptage se feront ***modulo*** n.
- Le décodage fonctionne grâce à une variante du ***petit théorème de Fermat***.

1 Division euclidienne et pgcd

1.1 Divisibilité et division euclidienne

Définition 1. Soient $a, b \in \mathbb{Z}$. On dit que b ***divise*** a et on note $b|a$ s'il existe $q \in \mathbb{Z}$ tel que

$$\boxed{a = bq}$$

.

Exemple 1. – $7|21$; $6|48$; a est pair si et seulement si $2|a$.
– Pour tout $a \in \mathbb{Z}$ on a $a|0$ et aussi $1|a$.
– Si $a|1$ alors $a = +1$ ou $a = -1$.
– $(a|b \text{ et } b|a) \implies b = \pm a$.
– $(a|b \text{ et } b|c) \implies a|c$.
– $(a|b \text{ et } a|c) \implies a|b+c$.

Théorème 1 (Division euclidienne).
Soit $a \in \mathbb{Z}$ et $b \in \mathbb{N} \setminus \{0\}$. Il ***existe*** des entiers $q, r \in \mathbb{Z}$ tels que

$$\boxed{a = bq + r \qquad \text{et} \qquad 0 \leqslant r < b}$$

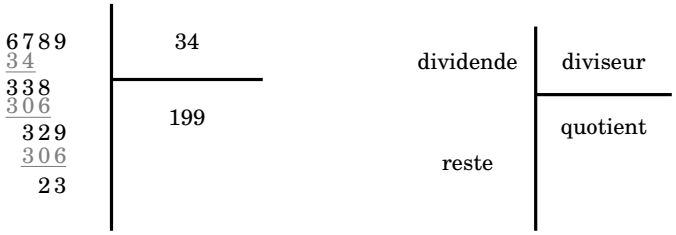
De plus q et r sont ***uniques***.

Nous avons donc l'équivalence : $r = 0$ si et seulement si b divise a.

Exemple 2. Pour calculer q et r on pose la division «classique». Si $a = 6789$ et $b = 34$ alors

$$6789 = 34 \times 199 + 23$$

On a bien $0 \leqslant 23 < 34$ (sinon c'est que l'on n'a pas été assez loin dans les calculs).



Démonstration. **Existence.** On peut supposer $a \geqslant 0$ pour simplifier. Soit $\mathcal{N} = \{n \in \mathbb{N} \mid bn \leqslant a\}$. C'est un ensemble non vide car $n = 0 \in \mathcal{N}$. De plus pour $n \in \mathcal{N}$, on a $n \leqslant a$. Il y a donc un nombre fini d'éléments dans $\mathcal{N}$, notons $q = \max \mathcal{N}$ le plus grand élément.
Alors $qb \leqslant a$ car $q \in \mathcal{N}$, et $(q+1)b > a$ car $q + 1 \notin \mathcal{N}$ donc

$$qb \leqslant a < (q+1)b = qb + b.$$

On définit alors $r = a - qb$, r vérifie alors $0 \leqslant r = a - qb < b$.
Unicité. Supposons que q', r' soient deux entiers qui vérifient les conditions du théorème. Tout d'abord $a = bq + r = bq' + r'$ et donc $b(q - q') = r' - r$. D'autre part $0 \leqslant r' < b$ et $0 \leqslant r < b$ donc $-b < r' - r < b$ (notez au passage la manipulation des inégalités). Mais $r' - r = b(q - q')$ donc on obtient $-b < b(q - q') < b$. On peut diviser par $b > 0$ pour avoir $-1 < q - q' < 1$. Comme $q - q'$ est un entier, la seule possibilité est $q - q' = 0$ et donc $q = q'$. Repartant de $r' - r = b(q - q')$ on obtient maintenant $r = r'$.

□

1.2 pgcd de deux entiers

Définition 2. Soient $a, b \in \mathbb{Z}$ deux entiers, non tous les deux nuls. Le plus grand entier qui divise à la fois a et b s'appelle le ***plus grand diviseur commun*** de a, b et se note $\text{pgcd}(a,b)$.

Exemple 3. – $\text{pgcd}(21,14) = 7$, $\text{pgcd}(12,32) = 4$, $\text{pgcd}(21,26) = 1$.
– $\text{pgcd}(a,ka) = a$, pour tout $k \in \mathbb{Z}$ et $a \geqslant 0$.
– Cas particuliers. Pour tout $a \geqslant 0$: $\text{pgcd}(a,0) = a$ et $\text{pgcd}(a,1) = 1$.

1.3 Algorithme d'Euclide

Lemme 1. Soient $a, b \in \mathbb{N}^*$. Écrivons la division euclidienne $a = bq + r$. Alors

$$\boxed{\text{pgcd}(a,b) = \text{pgcd}(b,r)}$$

En fait on a même $\text{pgcd}(a,b) = \text{pgcd}(b, a - qb)$ pour tout $q \in \mathbb{Z}$. Mais pour optimiser l'algorithme d'Euclide on applique le lemme avec q le quotient.

Démonstration. Nous allons montrer que les diviseurs de a et de b sont exactement les mêmes que les diviseurs de b et r. Cela impliquera le résultat car les plus grands diviseurs seront bien sûr les mêmes.
– Soit d un diviseur de a et de b. Alors d divise b donc aussi bq, en plus d divise a donc d divise $bq - a = r$.
– Soit d un diviseur de b et de r. Alors d divise aussi $bq + r = a$.

□

Algorithme d'Euclide.
On souhaite calculer le pgcd de $a, b \in \mathbb{N}^*$. On peut supposer $a \geqslant b$. On calcule des divisions euclidiennes successives. Le pgcd sera le dernier reste non nul.
– division de a par b, $a = bq_1 + r_1$. Par le lemme 1, $\text{pgcd}(a,b) = \text{pgcd}(b,r_1)$ et si $r_1 = 0$ alors $\text{pgcd}(a,b) = b$ sinon on continue :
– $b = r_1 q_2 + r_2$, $\quad \text{pgcd}(a,b) = \text{pgcd}(b,r_1) = \text{pgcd}(r_1,r_2)$,
– $r_1 = r_2 q_3 + r_3$, $\quad \text{pgcd}(a,b) = \text{pgcd}(r_2,r_3)$,
– ...
– $r_{k-2} = r_{k-1} q_k + r_k$, $\text{pgcd}(a,b) = \text{pgcd}(r_{k-1},r_k)$,
– $r_{k-1} = r_k q_k + 0$. $\text{pgcd}(a,b) = \text{pgcd}(r_k,0) = r_k$.
Comme à chaque étape le reste est plus petit que le quotient on sait que $0 \leqslant r_{i+1} < r_i$. Ainsi l'algorithme se termine car nous sommes sûr d'obtenir un reste nul, les restes formant une suite décroissante d'entiers positifs ou nuls : $b > r_1 > r_2 > \ldots \geqslant 0$

Exemple 4. Calculons le pgcd de $a = 600$ et $b = 124$.

$$\begin{array}{rcrcrcr} 600 & = & 124 & \times & 4 & + & 104 \\ 124 & = & 104 & \times & 1 & + & 20 \\ 104 & = & 20 & \times & 5 & + & 4 \\ 20 & = & 4 & \times & 5 & + & 0 \end{array}$$

Ainsi $\text{pgcd}(600,124) = 4$.

Voici un exemple plus compliqué :

Exemple 5. Calculons $\text{pgcd}(9945,3003)$.

$$\begin{array}{rcrcrcr} 9945 & = & 3003 & \times & 3 & + & 936 \\ 3003 & = & 936 & \times & 3 & + & 195 \\ 936 & = & 195 & \times & 4 & + & 156 \\ 195 & = & 156 & \times & 1 & + & 39 \\ 156 & = & 39 & \times & 4 & + & 0 \end{array}$$

Ainsi $\text{pgcd}(9945,3003) = 39$.

1.4 Nombres premiers entre eux

Définition 3. Deux entiers a, b sont ***premiers entre eux*** si $\text{pgcd}(a,b) = 1$.

Exemple 6. Pour tout $a \in \mathbb{Z}$, a et $a+1$ sont premiers entre eux. En effet soit d un diviseur commun à a et à $a+1$. Alors d divise aussi $a+1-a$. Donc d divise 1 mais alors $d = -1$ ou $d = +1$. Le plus grand diviseur de a et $a+1$ est donc 1. Et donc $\text{pgcd}(a, a+1) = 1$.

Si deux entiers ne sont pas premiers entre eux, on peut s'y ramener en divisant par leur pgcd :

Exemple 7. Pour deux entiers quelconques $a, b \in \mathbb{Z}$, notons $d = \text{pgcd}(a,b)$. La décomposition suivante est souvent utile :

$$\boxed{\begin{cases} a &= a'd \\ b &= b'd \end{cases} \quad \text{avec} \quad a', b' \in \mathbb{Z} \text{ et } \text{pgcd}(a', b') = 1}$$

1.5 Mini-exercices

1. Écrire la division euclidienne de 111 111 par $20xx$, où $20xx$ est l'année en cours.
2. Montrer qu'un diviseur positif de 10 008 et de 10 014 appartient nécessairement à $\{1, 2, 3, 6\}$.
3. Calculer $\text{pgcd}(560, 133)$, $\text{pgcd}(12\,121, 789)$, $\text{pgcd}(99\,999, 1110)$.
4. Trouver tous les entiers $1 \leqslant a \leqslant 50$ tels que a et 50 soient premiers entre eux. Même question avec 52.

2 Théorème de Bézout

2.1 Théorème de Bézout

Théorème 2 (Théorème de Bézout).
Soient a, b des entiers. Il existe des entiers $u, v \in \mathbb{Z}$ tels que

$$\boxed{au + bv = \text{pgcd}(a,b)}$$

La preuve découle de l'algorithme d'Euclide. Les entiers u, v ne sont pas uniques. Les entiers u, v sont des ***coefficients de Bézout***. Ils s'obtiennent en «remontant» l'algorithme d'Euclide.

Exemple 8. Calculons les coefficients de Bézout pour $a = 600$ et $b = 124$. Nous reprenons les calculs effectués pour trouver $\text{pgcd}(600, 124) = 4$. La partie gauche est l'algorithme d'Euclide. La partie droite s'obtient de *bas en haut*. On exprime le pgcd à l'aide de la dernière ligne où le reste est non nul. Puis on remplace le reste de la ligne précédente, et ainsi de suite jusqu'à arriver à la première ligne.

$$\begin{array}{rcrcrcl} 600 &=& 124 &\times& 4 &+& 104 \\ 124 &=& 104 &\times& 1 &+& 20 \\ 104 &=& 20 &\times& 5 &+& 4 \\ 20 &=& 4 &\times& 5 &+& 0 \end{array}$$

$$\begin{array}{rcl} 4 &=& 124 \times (-5) + (600 - 124 \times 4) \times 6 = 600 \times 6 + 124 \times (-29) \\ 4 &=& 104 - (124 - 104 \times 1) \times 5 = 124 \times (-5) + 104 \times 6 \\ 4 &=& 104 - 20 \times 5 \end{array}$$

Ainsi pour $u = 6$ et $v = -29$ alors $600 \times 6 + 124 \times (-29) = 4$.

Remarque. – Soignez vos calculs et leur présentation. C'est un algorithme : vous devez aboutir au bon résultat ! Dans la partie droite, il faut à chaque ligne bien la reformater. Par exemple $104 - (124 - 104 \times 1) \times 5$ se réécrit en $124 \times (-5) + 104 \times 6$ afin de pouvoir remplacer ensuite 104.
– N'oubliez de vérifier vos calculs ! C'est rapide et vous serez certain que vos calculs sont exacts. Ici on vérifie à la fin que $600 \times 6 + 124 \times (-29) = 4$.

Exemple 9. Calculons les coefficients de Bézout correspondant à $\text{pgcd}(9945, 3003) = 39$.

$$\begin{array}{rcrcrcr} 9945 & = & 3003 & \times & 3 & + & 936 \\ 3003 & = & 936 & \times & 3 & + & 195 \\ 936 & = & 195 & \times & 4 & + & 156 \\ 195 & = & 156 & \times & 1 & + & 39 \\ 156 & = & 39 & \times & 4 & + & 0 \end{array} \qquad \begin{array}{rcl} 39 & = & 9945 \times (-16) + 3003 \times 53 \\ 39 & = & \cdots \\ 39 & = & \cdots \\ 39 & = & 195 - 156 \times 1 \end{array}$$

À vous de finir les calculs. On obtient $9945 \times (-16) + 3003 \times 53 = 39$.

2.2 Corollaires du théorème de Bézout

Corollaire 1. Si $d|a$ et $d|b$ alors $d|\text{pgcd}(a,b)$.

Exemple : $4|16$ et $4|24$ donc 4 doit divisé $\text{pgcd}(16,24)$ qui effectivement vaut 8.

Démonstration. Comme $d|au$ et $d|bv$ donc $d|au+bv$. Par le théorème de Bézout $d|\text{pgcd}(a,b)$. □

Corollaire 2. Soient a, b deux entiers. a, b sont premiers entre eux **si et seulement si** il existe $u, v \in \mathbb{Z}$ tels que

$$\boxed{au + bv = 1}$$

Démonstration. Le sens $\Rightarrow$ est une conséquence du théorème de Bézout.
Pour le sens $\Leftarrow$ on suppose qu'il existe u, v tels que $au + bv = 1$. Comme $\text{pgcd}(a,b)|a$ alors $\text{pgcd}(a,b)|au$. De même $\text{pgcd}(a,b)|bv$. Donc $\text{pgcd}(a,b)|au+bv = 1$. Donc $\text{pgcd}(a,b) = 1$. □

Remarque. Si on trouve deux entiers u', v' tels que $au' + bv' = d$, cela n'implique **pas** que $d = \text{pgcd}(a,b)$. On sait seulement alors que $\text{pgcd}(a,b)|d$. Par exemple $a = 12$, $b = 8$; $12 \times 1 + 8 \times 3 = 36$ et $\text{pgcd}(a,b) = 4$.

Corollaire 3 (Lemme de Gauss). Soient $a, b, c \in \mathbb{Z}$.

$$\boxed{\text{Si} \quad a|bc \quad \text{et} \quad \text{pgcd}(a,b) = 1 \quad \text{alors} \quad a|c}$$

Exemple : si $4|7 \times c$, et comme 4 et 7 sont premiers entre eux, alors $4|c$.

Démonstration. Comme $\text{pgcd}(a,b) = 1$ alors il existe $u, v \in \mathbb{Z}$ tels que $au + bv = 1$. On multiplie cette égalité par c pour obtenir $acu + bcv = c$. Mais $a|acu$ et par hypothèse $a|bcv$ donc a divise $acu + bcv = c$. □

2.3 Équations $ax + by = c$

Proposition 1.
Considérons l'équation

$$ax + by = c \qquad \text{(E)}$$

où $a, b, c \in \mathbb{Z}$.

1. L'équation (E) possède des solutions $(x, y) \in \mathbb{Z}^2$ si et seulement si $\text{pgcd}(a,b)|c$.
2. Si $\text{pgcd}(a,b)|c$ alors il existe même une infinité de solutions entières et elles sont exactement les $(x, y) = (x_0 + \alpha k, y_0 + \beta k)$ avec $x_0, y_0, \alpha, \beta \in \mathbb{Z}$ fixés et k parcourant $\mathbb{Z}$.

Le premier point est une conséquence du théorème de Bézout. Nous allons voir sur un exemple comment prouver le second point et calculer explicitement les solutions. Il est bon de refaire toutes les étapes de la démonstration à chaque fois.

Exemple 10. Trouver les solutions entières de

$$161x + 368y = 115 \qquad \text{(E)}$$

– **Première étape. Y a-t'il de solutions ? L'algorithme d'Euclide.** On effectue l'algorithme d'Euclide pour calculer le pgcd de $a = 161$ et $b = 368$.

$$\begin{array}{rcrcrcr} 368 & = & 161 & \times & 2 & + & 46 \\ 161 & = & 46 & \times & 3 & + & 23 \\ 46 & = & 23 & \times & 2 & + & 0 \end{array}$$

Donc $\text{pgcd}(368, 161) = 23$. Comme $115 = 5 \times 23$ alors $\text{pgcd}(368, 161) | 115$. Par le théorème de Bézout, l'équation (E) admet des solutions entières.

– **Deuxième étape. Trouver une solution particulière : la remontée de l'algorithme d'Euclide.** On effectue la remontée de l'algorithme d'Euclide pour calculer les coefficients de Bézout.

$$\begin{array}{rcrcrcr} 368 & = & 161 & \times & 2 & + & 46 \\ 161 & = & 46 & \times & 3 & + & 23 \\ 46 & = & 23 & \times & 2 & + & 0 \end{array} \qquad \begin{array}{rcl} 23 & = & 161 + (368 - 2 \times 161) \times (-3) = 161 \times 7 + 368 \times (-3) \\ 23 & = & 161 - 3 \times 46 \end{array}$$

On trouve donc $161 \times 7 + 368 \times (-3) = 23$. Comme $115 = 5 \times 23$ en multipliant par 5 on obtient :

$$161 \times 35 + 368 \times (-15) = 115$$

Ainsi $(x_0, y_0) = (35, -15)$ est une ***solution particulière*** de (E).

– **Troisième étape. Recherche de toutes les solutions.** Soit $(x, y) \in \mathbb{Z}^2$ une solution de (E). Nous savons que (x_0, y_0) est aussi solution. Ainsi :

$$161x + 368y = 115 \quad \text{et} \quad 161x_0 + 368y_0 = 115$$

(on n'a aucun intérêt à remplacer x_0 et y_0 par leurs valeurs). La différence de ces deux égalités conduit à

$$\begin{array}{rl} & 161 \times (x - x_0) + 368 \times (y - y_0) = 0 \\ \Longrightarrow & 23 \times 7 \times (x - x_0) + 23 \times 16 \times (y - y_0) = 0 \\ \Longrightarrow & 7(x - x_0) = -16(y - y_0) \qquad (*) \end{array}$$

Nous avons simplifier par 23 qui est le pgcd de 161 et 368. (Attention, n'oubliez surtout pas cette simplification, sinon la suite du raisonnement serait fausse.)
Ainsi $7 | 16(y - y_0)$, or $\text{pgcd}(7, 16) = 1$ donc par le lemme de Gauss $7 | y - y_0$. Il existe donc $k \in \mathbb{Z}$ tel que $y - y_0 = 7 \times k$. Repartant de l'équation $(*)$: $7(x - x_0) = -16(y - y_0)$. On obtient maintenant $7(x - x_0) = -16 \times 7 \times k$. D'où $x - x_0 = -16k$. (C'est le même k pour x et pour y.) Nous avons donc $(x, y) = (x_0 - 16k, y_0 + 7k)$. Il n'est pas dur de voir que tout couple de cette forme est solution de l'équation (E). Il reste donc juste à substituer (x_0, y_0) par sa valeur et nous obtenons :

Les solutions entières de $161x + 368y = 115$ sont les $(x, y) = (35 - 16k, -15 + 7k)$, k parcourant $\mathbb{Z}$.

Pour se rassurer, prenez une valeur de k au hasard et vérifiez que vous obtenez bien une solution de l'équation.

2.4 ppcm

Définition 4. Le $\text{ppcm}(a, b)$ (***plus petit multiple commun***) est le plus petit entier $\geqslant 0$ divisible par a et par b.

Par exemple $\text{ppcm}(12, 9) = 36$.
Le pgcd et le ppcm sont liés par la formule suivante :

Proposition 2.
Si a, b sont des entiers (non tous les deux nuls) alors

$$\boxed{\text{pgcd}(a,b)\times\text{ppcm}(a,b)=|ab|}$$

Démonstration. Posons $d=\text{pgcd}(a,b)$ et $m=\frac{|ab|}{\text{pgcd}(a,b)}$. Pour simplifier on suppose $a>0$ et $b>0$. On écrit $a=da'$ et $b=db'$. Alors $ab=d^2a'b'$ et donc $m=da'b'$. Ainsi $m=ab'=a'b$ est un multiple de a et de b. Il reste à montrer que c'est le plus petit multiple. Si n est un autre multiple de a et de b alors $n=ka=\ell b$ donc $kda'=\ell db'$ et $ka'=\ell b'$. Or $\text{pgcd}(a',b')=1$ et $a'|\ell b'$ donc $a'|\ell$. Donc $a'b|\ell b$ et ainsi $m=a'b|\ell b=n$. □

Voici un autre résultat concernant le ppcm qui se démontre en utilisant la décomposition en facteurs premiers :

Proposition 3.
Si $a|c$ et $b|c$ alors $\text{ppcm}(a,b)|c$.

Il serait faux de penser que $ab|c$. Par exemple $6|36$, $9|36$ mais 6×9 ne divise pas 36. Par contre $\text{ppcm}(6,9)=18$ divise bien 36.

2.5 Mini-exercices

1. Calculer les coefficients de Bézout correspondant à $\text{pgcd}(560,133)$, $\text{pgcd}(12\,121,789)$.
2. Montrer à l'aide d'un corollaire du théorème de Bézout que $\text{pgcd}(a,a+1)=1$.
3. Résoudre les équations : $407x+129y=1$; $720x+54y=6$; $216x+92y=8$.
4. Trouver les couples (a,b) vérifiant $\text{pgcd}(a,b)=12$ et $\text{ppcm}(a,b)=360$.

3 Nombres premiers

Les nombres premiers sont –en quelque sorte– les briques élémentaires des entiers : tout entier s'écrit comme produit de nombres premiers.

3.1 Une infinité de nombres premiers

Définition 5. Un ***nombre premier*** p est un entier $\geqslant 2$ dont les seuls diviseurs positifs sont 1 et p.

Exemples : $2,3,5,7,11$ sont premiers, $4=2\times 2$, $6=2\times 3$, $8=2\times 4$ ne sont pas premiers.

Lemme 2. Tout entier $n\geqslant 2$ admet un diviseur qui est un nombre premier.

Démonstration. Soit $\mathscr{D}$ l'ensemble des diviseurs de n qui sont $\geqslant 2$:

$$\mathscr{D}=\{k\geqslant 2 \mid \ k|n\}.$$

L'ensemble $\mathscr{D}$ est non vide (car $n\in\mathscr{D}$), notons alors $p=\min\mathscr{D}$.
Supposons, par l'absurde, que p ne soit pas un nombre premier alors p admet un diviseur q tel que $1<q<p$ mais alors q est aussi un diviseur de n et donc $q\in\mathscr{D}$ avec $q<p$. Ce qui donne une contradiction car p est le minimum. Conclusion : p est un nombre premier. Et comme $p\in\mathscr{D}$, p divise n. □

Proposition 4.
Il existe une infinité de nombres premiers.

Démonstration. Par l'absurde, supposons qu'il n'y ait qu'un nombre fini de nombres premiers que l'on note $p_1=2$, $p_2=3$, $p_3,\ldots,p_n$. Considérons l'entier $N=p_1\times p_2\times\cdots\times p_n+1$. Soit p un diviseur premier de N (un tel p existe par le lemme précédent), alors d'une part p est l'un des entiers p_i donc $p|p_1\times\cdots\times p_n$, d'autre part $p|N$ donc p divise la différence $N-p_1\times\cdots\times p_n=1$. Cela implique que $p=1$, ce qui contredit que p soit un nombre premier.
Cette contradiction nous permet de conclure qu'il existe une infinité de nombres premiers. □

3.2 Eratosthène et Euclide

Comment trouver les nombres premiers ? Le ***crible d'Eratosthène*** permet de trouver les premiers nombres premiers. Pour cela on écrit les premiers entiers : pour notre exemple de 2 à 25.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Rappelons-nous qu'un diviseur positif d'un entier n est inférieur ou égal à n. Donc 2 ne peut avoir comme diviseurs que 1 et 2 et est donc premier. On entoure 2. Ensuite on raye (ici en grisé) tous les multiples suivants de 2 qui ne seront donc pas premiers (car divisible par 2) :

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Le premier nombre restant de la liste est 3 et est nécessairement premier : il n'est pas divisible par un diviseur plus petit (sinon il serait rayé). On entoure 3 et on raye tous les multiples de 3 (6, 9, 12, ...).

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Le premier nombre restant est 5 et est donc premier. On raye les multiples de 5.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

7 est donc premier, on raye les multiples de 7 (ici pas de nouveaux nombres à barrer). Ainsi de suite : $11, 13, 17, 19, 23$ sont premiers.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Remarque. Si un nombre n n'est pas premier alors un de ses facteurs est $\leqslant \sqrt{n}$. En effet si $n = a \times b$ avec $a, b \geqslant 2$ alors $a \leqslant \sqrt{n}$ ou $b \leqslant \sqrt{n}$ (réfléchissez par l'absurde !). Par exemple pour tester si un nombre $\leqslant 100$ est premier il suffit de tester les diviseurs $\leqslant 10$. Et comme il suffit de tester les diviseurs premiers, il suffit en fait de tester la divisibilité par $2, 3, 5$ et 7. Exemple : 89 n'est pas divisible par $2, 3, 5, 7$ et est donc un nombre premier.

Proposition 5 (Lemme d'Euclide).
Soit p un nombre premier. Si $p|ab$ alors $p|a$ ou $p|b$.

Démonstration. Si p ne divise pas a alors p et a sont premiers entre eux (en effet les diviseurs de p sont 1 et p, mais seul 1 divise aussi a, donc $\text{pgcd}(a, p) = 1$). Ainsi par le lemme de Gauss $p|b$. □

Exemple 11. Si p est un nombre premier, $\sqrt{p}$ n'est pas un nombre rationnel.
La preuve se fait par l'absurde : écrivons $\sqrt{p} = \frac{a}{b}$ avec $a \in \mathbb{Z}, b \in \mathbb{N}^*$ et $\text{pgcd}(a, b) = 1$. Alors $p = \frac{a^2}{b^2}$ donc $pb^2 = a^2$. Ainsi $p|a^2$ donc par le lemme d'Euclide $p|a$. On peut alors écrire $a = pa'$ avec a' un entier. De l'équation $pb^2 = a^2$ on tire alors $b^2 = pa'^2$. Ainsi $p|b^2$ et donc $p|b$. Maintenant $p|a$ et $p|b$ donc a et b ne sont pas premiers entre eux. Ce qui contredit $\text{pgcd}(a, b) = 1$. Conclusion $\sqrt{p}$ n'est pas rationnel.

3.3 Décomposition en facteurs premiers

Théorème 3.
Soit $n \geqslant 2$ un entier. Il existe des nombres premiers $p_1 < p_2 < \ldots < p_r$ et des exposants entiers $\alpha_1, \alpha_2, \ldots, \alpha_r \geqslant 1$ tels que :

$$n = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_r^{\alpha_r}.$$

De plus les p_i et les α_i $(i = 1, \ldots, r)$ sont uniques.

Exemple : $24 = 2^3 \times 3$ est la décomposition en facteurs premiers. Par contre $36 = 2^2 \times 9$ n'est pas la décomposition en facteurs premiers c'est $2^2 \times 3^2$.

Remarque. La principale raison pour laquelle on choisit de dire que 1 n'est pas un nombre premier, c'est que sinon il n'y aurait plus unicité de la décomposition : $24 = 2^3 \times 3 = 1 \times 2^3 \times 3 = 1^2 \times 2^3 \times 3 = \cdots$

Démonstration. **Existence.** Nous allons démontrer l'existence de la décomposition par une récurrence sur n.
L'entier $n = 2$ est déjà décomposé. Soit $n \geqslant 3$, supposons que tout entier $< n$ admette une décomposition en facteurs premiers. Notons p_1 le plus petit nombre premier divisant n (voir le lemme 2). Si n est un nombre premier alors $n = p_1$ et c'est fini. Sinon on définit l'entier $n' = \frac{n}{p_1} < n$ et on applique notre hypothèse de récurrence à n' qui admet une décomposition en facteurs premiers. Alors $n = p_1 \times n'$ admet aussi une décomposition.

Unicité. Nous allons démontrer qu'une telle décomposition est unique en effectuant cette fois une récurrence sur la somme des exposants $\sigma = \sum_{i=1}^{r} \alpha_i$.
Si $\sigma = 1$ cela signifie $n = p_1$ qui est bien l'unique écriture possible.
Soit $\sigma \geqslant 2$. On suppose que les entiers dont la somme des exposants est $< \sigma$ ont une unique décomposition. Soit n un entier dont la somme des exposants vaut σ. Écrivons le avec deux décompositions :

$$n = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_r^{\alpha_r} = q_1^{\beta_1} \times q_2^{\beta_2} \times \cdots \times q_s^{\beta_s}.$$

(On a $p_1 < p_2 < \cdots$ et $q_1 < q_2 < \cdots$.)
Si $p_1 < q_1$ alors $p_1 < q_j$ pour tous les $j = 1, \ldots, s$. Ainsi p_1 divise $p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_r^{\alpha_r} = n$ mais ne divise pas $q_1^{\beta_1} \times q_2^{\beta_2} \times \cdots \times q_s^{\beta_s} = n$. Ce qui est absurde. Donc $p_1 \geqslant q_1$.
Si $p_1 > q_1$ un même raisonnement conduit aussi à une contradiction. On conclut que $p_1 = q_1$. On pose alors

$$n' = \frac{n}{p_1} = p_1^{\alpha_1 - 1} \times p_2^{\alpha_2} \times \cdots \times p_r^{\alpha_r} = q_1^{\beta_1 - 1} \times q_2^{\beta_2} \times \cdots \times q_s^{\beta_s}$$

L'hypothèse de récurrence qui s'applique à n' implique que ces deux décompositions sont les mêmes. Ainsi $r = s$ et $p_i = q_i$, $\alpha_i = \beta_i$, $i = 1, \ldots, r$. □

Exemple 12.

$$504 = 2^3 \times 3^2 \times 7, \quad 300 = 2^2 \times 3 \times 5^2.$$

Pour calculer le pgcd on réécrit ces décompositions :

$$504 = 2^3 \times 3^2 \times 5^0 \times 7^1, \quad 300 = 2^2 \times 3^1 \times 5^2 \times 7^0.$$

Le pgcd est le nombre obtenu en prenant le plus petit exposant de chaque facteur premier :

$$\text{pgcd}(504, 300) = 2^2 \times 3^1 \times 5^0 \times 7^0 = 12.$$

Pour le ppcm on prend le plus grand exposant de chaque facteur premier :

$$\text{ppcm}(504, 300) = 2^3 \times 3^2 \times 5^2 \times 7^1 = 12\,600$$

3.4 Mini-exercices

1. Montrer que $n! + 1$ n'est divisible par aucun des entiers $1, \ldots, n$. Est-ce toujours un nombre premier ?
2. Trouver tous les nombres premiers $\leqslant 103$.
3. Décomposer $a = 2\,340$ et $b = 15\,288$ en facteurs premiers. Calculer leur pgcd et leur ppcm.
4. Décomposer $48\,400$ en produit de facteurs premiers. Combien $48\,400$ admet-il de diviseurs ?
5. Soient $a, b \geqslant 0$. À l'aide de la décomposition en facteurs premiers, reprouver la formule $\text{pgcd}(a, b) \times \text{ppcm}(a, b) = a \times b$.

4 Congruences

4.1 Définition

Définition 6. Soit $n \geqslant 2$ un entier. On dit que a est ***congru*** à b ***modulo*** n, si n divise $b-a$. On note alors

$$a \equiv b \pmod{n}.$$

On note aussi parfois $a = b \pmod{n}$ ou $a \equiv b[n]$. Une autre formulation est

$$a \equiv b \pmod{n} \iff \exists k \in \mathbb{Z} \quad a = b + kn.$$

Remarquez que n divise a si et seulement si $a \equiv 0 \pmod{n}$.

Proposition 6.
1. La relation «congru modulo n» est une relation d'équivalence :
   - $a \equiv a \pmod{n}$,
   - si $a \equiv b \pmod{n}$ alors $b \equiv a \pmod{n}$,
   - si $a \equiv b \pmod{n}$ et $b \equiv c \pmod{n}$ alors $a \equiv c \pmod{n}$.
2. Si $a \equiv b \pmod{n}$ et $c \equiv d \pmod{n}$ alors $a + c \equiv b + d \pmod{n}$.
3. Si $a \equiv b \pmod{n}$ et $c \equiv d \pmod{n}$ alors $a \times c \equiv b \times d \pmod{n}$.
4. Si $a \equiv b \pmod{n}$ alors pour tout $k \geqslant 0$, $a^k \equiv b^k \pmod{n}$.

Exemple 13.
- $15 \equiv 1 \pmod{7}$, $72 \equiv 2 \pmod{7}$, $3 \equiv -11 \pmod{7}$,
- $5x + 8 \equiv 3 \pmod{5}$ pour tout $x \in \mathbb{Z}$,
- $11^{20xx} \equiv 1^{20xx} \equiv 1 \pmod{10}$, où $20xx$ est l'année en cours.

Démonstration.
1. Utiliser la définition.
2. Idem.
3. Prouvons la propriété multiplicative : $a \equiv b \pmod{n}$ donc il existe $k \in \mathbb{Z}$ tel que $a = b + kn$ et $c \equiv d \pmod{n}$ donc il existe $\ell \in \mathbb{Z}$ tel que $c \equiv d + \ell n$. Alors $a \times c = (b + kn) \times (d + \ell n) = bd + (b\ell + dk + k\ell n)n$ qui est bien de la forme $bd + mn$ avec $m \in \mathbb{Z}$. Ainsi $ac \equiv bd \pmod{n}$.
4. C'est une conséquence du point précédent : avec $a = c$ et $b = d$ on obtient $a^2 \equiv b^2 \pmod{n}$. On continue par récurrence.

□

Exemple 14. Critère de divisibilité par 9.

N est divisible par 9 si et seulement si la somme de ses chiffres est divisible par 9.

Pour prouver cela nous utilisons les congruences. Remarquons d'abord que $9|N$ équivaut à $N \equiv 0 \pmod{9}$ et notons aussi que $10 \equiv 1 \pmod{9}$, $10^2 \equiv 1 \pmod{9}$, $10^3 \equiv 1 \pmod{9}$,...
Nous allons donc calculer N modulo 9. Écrivons N en base 10 : $N = \underline{a_k \cdots a_2 a_1 a_0}$ (a_0 est le chiffre des unités, a_1 celui des dizaines,...) alors $N = 10^k a_k + \cdots + 10^2 a_2 + 10^1 a_1 + a_0$. Donc

$$\begin{aligned} N &= 10^k a_k + \cdots + 10^2 a_2 + 10^1 a_1 + a_0 \\ &\equiv a_k + \cdots + a_2 + a_1 + a_0 \pmod{9} \end{aligned}$$

Donc N est congru à la somme de ses chiffres modulo 9. Ainsi $N \equiv 0 \pmod{9}$ si et seulement si la somme des chiffres vaut 0 modulo 9.

Voyons cela sur un exemple : $N = 488889$. Ici $a_0 = 9$ est le chiffre des unités, $a_1 = 8$ celui des dizaines,... Cette écriture décimale signifie $N = 4\cdot 10^5 + 8\cdot 10^4 + 8\cdot 10^3 + 8\cdot 10^2 + 8\cdot 10 + 9$.

$$\begin{aligned} N &= 4\cdot 10^5 + 8\cdot 10^4 + 8\cdot 10^3 + 8\cdot 10^2 + 8\cdot 10 + 9 \\ &\equiv 4+8+8+8+8+9 \pmod 9 \\ &\equiv 45 \pmod 9 \qquad \text{et on refait la somme des chiffres de 45} \\ &\equiv 9 \pmod 9 \\ &\equiv 0 \pmod 9 \end{aligned}$$

Ainsi nous savons que 488889 est divisible par 9 sans avoir effectué de division euclidienne.

Remarque. Pour trouver un «bon» représentant de $a \pmod n$ on peut aussi faire la division euclidienne de a par n : $a = bn + r$ alors $a \equiv r \pmod n$ et $0 \leqslant r < n$.

Exemple 15. Les calculs bien menés avec les congruences sont souvent très rapides. Par exemple on souhaite calculer $2^{21} \pmod{37}$ (plus exactement on souhaite trouver $0 \leqslant r < 37$ tel que $2^{21} \equiv r \pmod{37}$). Plusieurs méthodes :

1. On calcule 2^{21}, puis on fait la division euclidienne de 2^{21} par 37, le reste est notre résultat. C'est laborieux !
2. On calcule successivement les 2^k modulo 37 : $2^1 \equiv 2 \pmod{37}$, $2^2 \equiv 4 \pmod{37}$, $2^3 \equiv 8 \pmod{37}$, $2^4 \equiv 16 \pmod{37}$, $2^5 \equiv 32 \pmod{37}$. Ensuite on n'oublie pas d'utiliser les congruences : $2^6 \equiv 64 \equiv 27 \pmod{37}$. $2^7 \equiv 2\cdot 2^6 \equiv 2\cdot 27 \equiv 54 \equiv 17 \pmod{37}$ et ainsi de suite en utilisant le calcul précédent à chaque étape. C'est assez efficace et on peut raffiner : par exemple on trouve $2^8 \equiv 34 \pmod{37}$ mais donc aussi $2^8 \equiv -3 \pmod{37}$ et donc $2^9 \equiv 2\cdot 2^8 \equiv 2\cdot(-3) \equiv -6 \equiv 31 \pmod{37}$,...
3. Il existe une méthode encore plus efficace : on écrit l'exposant 21 en base 2 : $21 = 2^4 + 2^2 + 2^0 = 16 + 4 + 1$. Alors $2^{21} = 2^{16}\cdot 2^4 \cdot 2^1$. Et il est facile de calculer successivement chacun de ces termes car les exposants sont des puissances de 2. Ainsi $2^8 \equiv (2^4)^2 \equiv 16^2 \equiv 256 \equiv 34 \equiv -3 \pmod{37}$ et $2^{16} \equiv \left(2^8\right)^2 \equiv (-3)^2 \equiv 9 \pmod{37}$. Nous obtenons $2^{21} \equiv 2^{16}\cdot 2^4 \cdot 2^1 \equiv 9\times 16 \times 2 \equiv 288 \equiv 29 \pmod{37}$.

4.2 Équation de congruence $ax \equiv b \pmod n$

Proposition 7.
Soit $a \in \mathbb{Z}^*$, $b \in \mathbb{Z}$ fixés et $n \geqslant 2$. Considérons l'équation $\boxed{ax \equiv b \pmod n}$ d'inconnue $x \in \mathbb{Z}$:

1. Il existe des solutions si et seulement si $\operatorname{pgcd}(a,n) | b$.
2. Les solutions sont de la forme $x = x_0 + \ell \frac{n}{\operatorname{pgcd}(a,n)}$, $\ell \in \mathbb{Z}$ où x_0 est une solution particulière. Il existe donc $\operatorname{pgcd}(a,n)$ classes de solutions.

Exemple 16. Résolvons l'équation $9x \equiv 6 \pmod{24}$. Comme $\operatorname{pgcd}(9,24) = 3$ divise 6 la proposition ci-dessus nous affirme qu'il existe des solutions. Nous allons les calculer. (Il est toujours préférable de refaire rapidement les calculs que d'apprendre la formule). Trouver x tel que $9x \equiv 6 \pmod{24}$ est équivalent à trouver x et k tels que $9x = 6 + 24k$. Mis sous la forme $9x - 24k = 6$ il s'agit alors d'une équation que nous avons étudier en détails (voir section 2.3). Il y a bien des solutions car $\operatorname{pgcd}(9,24) = 3$ divise 6. En divisant par le pgcd on obtient l'équation équivalente :

$$3x - 8k = 2.$$

Pour le calcul du pgcd et d'une solution particulière nous utilisons normalement l'algorithme d'Euclide et sa remontée. Ici il est facile de trouver une solution particulière ($x_0 = 6, k_0 = 2$) à la main.
On termine comme pour les équations de la section 2.3. Si (x,k) est une solution de $3x - 8k = 2$ alors par soustraction on obtient $3(x - x_0) - 8(k - k_0) = 0$ et on trouve $x = x_0 + 8\ell$, avec $\ell \in \mathbb{Z}$ (le terme k ne nous intéresse pas). Nous avons donc trouvé les x qui sont solutions de $3x - 8k = 2$, ce qui équivaut à $9x - 24k = 6$, ce qui équivaut encore à $9x \equiv 6 \pmod{24}$. Les solutions sont de la forme $x = 6 + 8\ell$. On préfère les regrouper en 3 classes modulo 24 :

$$x_1 = 6 + 24m, \quad x_2 = 14 + 24m, \quad x_3 = 22 + 24m \qquad \text{avec } m \in \mathbb{Z}.$$

Remarque. Expliquons le terme de «classe» utilisé ici. Nous avons considérer ici que l'équation $9x \equiv 6 \pmod{24}$ est une équation d'entiers. On peut aussi considérer que $9, x, 6$ sont des classes d'équivalence modulo 24, et l'on noterait alors $\overline{9}\overline{x} = \overline{6}$. On trouverait comme solutions trois classes d'équivalence :

$$\overline{x_1} = \overline{6}, \quad \overline{x_2} = \overline{14}, \quad \overline{x_3} = \overline{22}.$$

Démonstration. 1.

$$\begin{aligned} & x \in \mathbb{Z} \text{ est un solution de l'équation } ax \equiv b \pmod{n} \\ \iff & \exists k \in \mathbb{Z} \quad ax = b + kn \\ \iff & \exists k \in \mathbb{Z} \quad ax - kn = b \\ \iff & \operatorname{pgcd}(a,n) | b \quad \text{par la proposition 1} \end{aligned}$$

Nous avons juste transformé notre équation $ax \equiv b \pmod{n}$ en une équation $ax - kn = b$ étudiée auparavant (voir section 2.3), seules les notations changent : $au + bv = c$ devient $ax - kn = b$.

2. Supposons qu'il existe des solutions. Nous allons noter $d = \operatorname{pgcd}(a,n)$ et écrire $a = da'$, $n = dn'$ et $b = db'$ (car par le premier point $d|b$). L'équation $ax - kn = b$ d'inconnues $x, k \in \mathbb{Z}$ est alors équivalente à l'équation $a'x - kn' = b'$, notée $(\star)$. Nous savons résoudre cette équation (voir de nouveau la proposition 1), si (x_0, k_0) est une solution particulière de $(\star)$ alors on connaît tous les (x,k) solutions. En particulier $x = x_0 + \ell n'$ avec $\ell \in \mathbb{Z}$ (les k ne nous intéressent pas ici).
Ainsi les solutions $x \in \mathbb{Z}$ sont de la forme $x = x_0 + \ell \frac{n}{\operatorname{pgcd}(a,n)}$, $\ell \in \mathbb{Z}$ où x_0 est une solution particulière de $ax \equiv b \pmod{n}$. Et modulo n cela donne bien $\operatorname{pgcd}(a,n)$ classes distinctes.

□

4.3 Petit théorème de Fermat

Théorème 4 (Petit théorème de Fermat).
Si p est un nombre premier et $a \in \mathbb{Z}$ alors

$$\boxed{a^p \equiv a \pmod{p}}$$

Corollaire 4. Si p ne divise pas a alors

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

Lemme 3. p divise $\binom{p}{k}$ pour $1 \leqslant k \leqslant p-1$, c'est-à-dire $\binom{p}{k} \equiv 0 \pmod{p}$.

Démonstration. $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ donc $p! = k!(p-k)!\binom{p}{k}$. Ainsi $p | k!(p-k)!\binom{p}{k}$. Or comme $1 \leqslant k \leqslant p-1$ alors p ne divise pas $k!$ (sinon p divise l'un des facteurs de $k!$ mais il sont tous $< p$). De même p ne divise pas $(p-k)!$, donc par le lemme d'Euclide p divise $\binom{p}{k}$. □

Preuve du théorème. Nous le montrons par récurrence pour les $a \geqslant 0$.
- Si $a = 0$ alors $0 \equiv 0 \pmod{p}$.
- Fixons $a \geqslant 0$ et supposons que $a^p \equiv a \pmod{p}$. Calculons $(a+1)^p$ à l'aide de la formule du binôme de Newton :

$$(a+1)^p = a^p + \binom{p}{p-1} a^{p-1} + \binom{p}{p-2} a^{p-2} + \cdots + \binom{p}{1} + 1$$

Réduisons maintenant modulo p :

$$\begin{aligned} (a+1)^p &\equiv a^p + \binom{p}{p-1} a^{p-1} + \binom{p}{p-2} a^{p-2} + \cdots + \binom{p}{1} + 1 \pmod{p} \\ &\equiv a^p + 1 \pmod{p} \quad \text{grâce au lemme 3} \\ &\equiv a + 1 \pmod{p} \quad \text{à cause de l'hypothèse de récurrence} \end{aligned}$$

– Par le principe de récurrence nous avons démontré le petit théorème de Fermat pour tout $a \geqslant 0$. Il n'est pas dur d'en déduire le cas des $a \leqslant 0$.

□

Exemple 17. Calculons $14^{3141} \pmod{17}$. Le nombre 17 étant premier on sait par le petit théorème de Fermat que $14^{16} \equiv 1 \pmod{17}$. Écrivons la division euclidienne de 3141 par 16 :

$$3141 = 16 \times 196 + 5.$$

Alors

$$14^{3141} \equiv 14^{16\times 196+5} \equiv 14^{16\times 196} \times 14^5 \equiv \left(14^{16}\right)^{196} \times 14^5 \equiv 1^{196} \times 14^5 \equiv 14^5 \pmod{17}$$

Il ne reste plus qu'à calculer 14^5 modulo 17. Cela peut se faire rapidement : $14 \equiv -3 \pmod{17}$ donc $14^2 \equiv (-3)^2 \equiv 9 \pmod{17}$, $14^3 \equiv 14^2 \times 14 \equiv 9 \times (-3) \equiv -27 \equiv 7 \pmod{17}$, $14^5 \equiv 14^2 \times 14^3 \equiv 9 \times 7 \equiv 63 \equiv 12 \pmod{17}$. Conclusion : $14^{3141} \equiv 14^5 \equiv 12 \pmod{17}$.

4.4 Mini-exercices

1. Calculer les restes modulo 10 de $122 + 455$, 122×455, 122^{455}. Mêmes calculs modulo 11, puis modulo 12.
2. Prouver qu'un entier est divisible par 3 si et seulement si la somme de ses chiffres est divisible par 3.
3. Calculer $3^{10} \pmod{23}$.
4. Calculer $3^{100} \pmod{23}$.
5. Résoudre les équations $3x \equiv 4 \pmod{7}$, $4x \equiv 14 \pmod{30}$.

Auteurs
Arnaud Bodin
Benjamin Boutin
Pascal Romon

Cryptographie

Vidéo ■ partie 1. Le chiffrement de César
Vidéo ■ partie 2. Le chiffrement de Vigenère

1 Le chiffrement de César

1.1 César a dit...

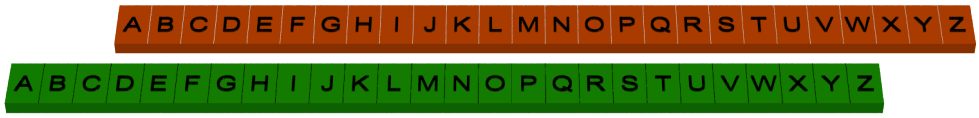
Jules César a-t-il vraiment prononcé la célèbre phrase :

DOHD MDFWD HVW

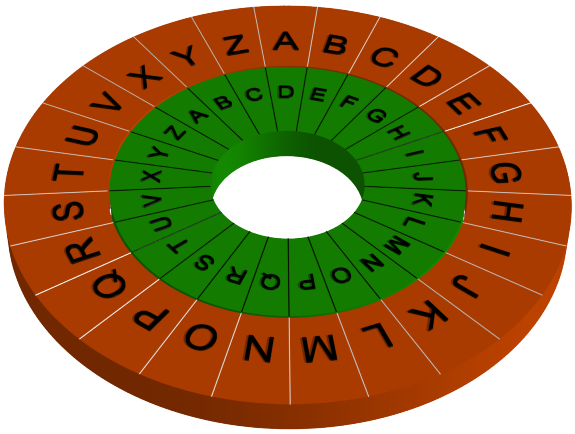
ou bien comme le disent deux célèbres Gaulois : « Ils sont fous ces romains ! ».
En fait César, pour ses communications importantes à son armée, cryptait ses messages. Ce que l'on appelle le chiffrement de César est un décalage des lettres : pour crypter un message, A devient D, B devient E, C devient F,...

$$A \longmapsto D \quad B \longmapsto E \quad C \longmapsto F \quad \dots \qquad W \longmapsto Z \quad X \longmapsto A \quad Y \longmapsto B \quad Z \longmapsto C$$

Voici une figure avec l'alphabet d'origine en haut et en **rouge**, en correspondance avec l'alphabet pour le chiffrement en-dessous et en **vert**.



Nous adopterons la convention suivante, en **vert** c'est la partie du message à laquelle tout le monde a accès (ou qui pourrait être intercepté), c'est donc le message crypté. Alors qu'en **rouge** c'est la partie du message confidentiel, c'est le message en clair.
Pour prendre en compte aussi les dernières lettres de l'alphabet, il est plus judicieux de représenté l'alphabet sur un anneau. Ce décalage est un ***décalage circulaire*** sur les lettres de l'alphabet.



Pour déchiffrer le message de César, il suffit de décaler les lettres dans l'autre sens, D se déchiffre en A, E en B,...
Et la célèbre phrase de César est :

ALEA JACTA EST

qui traduite du latin donne « Les dés sont jetés ».

1.2 Des chiffres et des lettres

Il est plus facile de manipuler des nombres que des lettres, aussi nous passons à une formulation mathématique. Nous associons à chacune des 26 lettres de A à Z un nombre de 0 à 25. En termes mathématiques, nous définissons une bijection :

$$f : \{A, B, C, \dots, Z\} \longrightarrow \{0, 1, 2, \dots, 25\}$$

par

$$A \longmapsto 0 \quad B \longmapsto 1 \quad C \longmapsto 2 \quad \dots \quad Z \longmapsto 25$$

Ainsi "**A L E A**" devient "**0 11 4 0**".
Le chiffrement de César est un cas particulier de ***chiffrement mono-alphabétique***, c'est-à-dire un chiffrement lettre à lettre.
Quel est l'intérêt ? Nous allons voir que le chiffrement de César correspond à une opération mathématique très simple. Pour cela, rappelons la notion de congruence et l'ensemble $\mathbb{Z}/26\mathbb{Z}$.

1.3 Modulo

Soit $n \geqslant 2$ un entier fixé.

Définition 7. On dit que a ***est congru à*** b ***modulo*** n, si n divise $b-a$. On note alors

$$a \equiv b \pmod{n}.$$

Pour nous $n = 26$. Ce qui fait que $28 \equiv 2 \pmod{26}$, car $28-2$ est bien divisible par 26. De même $85 = 3\times 26+7$ donc $85 \equiv 7 \pmod{26}$.
On note $\mathbb{Z}/26\mathbb{Z}$ l'ensemble de tous les éléments de $\mathbb{Z}$ modulo 26. Cet ensemble peut par exemple être représenté par les 26 éléments $\{0,1,2,\dots,25\}$. En effet, puisqu'on compte modulo 26 :

$$0,\ 1,\ 2,\ \dots,\ 25, \quad \text{puis} \quad 26 \equiv 0,\ 27 \equiv 1,\ 28 \equiv 2, \dots, \quad 52 \equiv 0,\ 53 \equiv 1, \dots$$

et de même $-1 \equiv 25$, $-2 \equiv 24$,...

Plus généralement $\mathbb{Z}/n\mathbb{Z}$ contient n éléments. Pour un entier $a \in \mathbb{Z}$ quelconque, son ***représentant*** dans $\{0,1,2,\dots,n-1\}$ s'obtient comme le reste k de la division euclidienne de a par n : $a = bn+k$. De sorte que $a \equiv k \pmod{n}$ et $0 \leqslant k < n$.

De façon naturelle l'addition et la multiplication d'entiers se transposent dans $\mathbb{Z}/n\mathbb{Z}$.

Pour $a, b \in \mathbb{Z}/n\mathbb{Z}$, on associe $a+b \in \mathbb{Z}/n\mathbb{Z}$.
Par exemple dans $\mathbb{Z}/26\mathbb{Z}$, $15+13$ égale 2. En effet $15+13 = 28 \equiv 2 \pmod{26}$. Autre exemple : que vaut $133+64$? $133+64 = 197 = 7\times 26+15 \equiv 15 \pmod{26}$. Mais on pourrait procéder différemment : tout d'abord $133 = 5\times 26+3 \equiv 3 \pmod{26}$ et $64 = 2\times 26+12 \equiv 12 \pmod{26}$. Et maintenant sans calculs : $133+64 \equiv 3+12 \equiv 15 \pmod{26}$.

On fait de même pour la multiplication : pour $a, b \in \mathbb{Z}/n\mathbb{Z}$, on associe $a \times b \in \mathbb{Z}/n\mathbb{Z}$.
Par exemple 3×12 donne 10 modulo 26, car $3\times 12 = 36 = 1\times 26+10 \equiv 10 \pmod{26}$. De même : $3\times 27 = 81 = 3\times 26+3 \equiv 3 \pmod{26}$. Une autre façon de voir la même opération est d'écrire d'abord $27 = 1 \pmod{26}$ puis $3\times 27 \equiv 3\times 1 \equiv 3 \pmod{26}$.

1.4 Chiffrer et déchiffrer

Le chiffrement de César est simplement une addition dans $\mathbb{Z}/26\mathbb{Z}$! Fixons un entier k qui est le décalage (par exemple $k = 3$ dans l'exemple de César ci-dessus) et définissons la ***fonction de chiffrement de César de décalage*** k qui va de l'ensemble $\mathbb{Z}/26\mathbb{Z}$ dans lui-même :

$$C_k : \left\{ \begin{array}{ccc} \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto & x+k \end{array} \right.$$

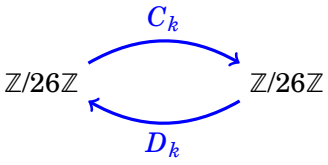
Par exemple, pour $k = 3$: $C_3(0) = 3$, $C_3(1) = 4\dots$
Pour déchiffrer, rien de plus simple ! Il suffit d'aller dans l'autre sens, c'est-à-dire ici de soustraire. La ***fonction de déchiffrement de César de décalage*** k est

$$D_k : \left\{ \begin{array}{ccc} \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto & x-k \end{array} \right.$$

En effet, si 1 a été chiffré en 4, par la fonction C_3 alors $D_3(4) = 4-3 = 1$. On retrouve le nombre original. Mathématiquement, D_k est la bijection réciproque de C_k, ce qui implique que pour tout $x \in \mathbb{Z}/26\mathbb{Z}$:

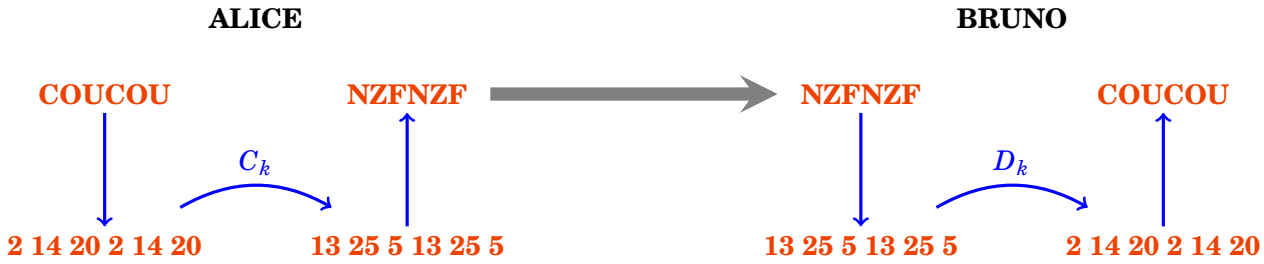
$$D_k\big(C_k(x)\big) = x$$

En d'autres termes, si x est un nombre, on applique la fonction de chiffrement pour obtenir le nombre crypté $y = C_k(x)$; ensuite la fonction de déchiffrement fait bien ce que l'on attend d'elle $D_k(y) = x$, on retrouve le nombre original x.



Une autre façon de voir la fonction de déchiffrement est de remarquer que $D_k(x) = C_{-k}(x)$. Par exemple $C_{-3}(x) = x + (-3) \equiv x + 23 \pmod{26}$.

Voici le principe du chiffrement : Alice veut envoyer des messages secrets à Bruno. Ils se sont d'abord mis d'accord sur une clé secrète k, par exemple $k = 11$. Alice veut envoyer le message "**COUCOU**" à Bruno. Elle transforme "**COUCOU**" en "**2 14 20 2 14 20**". Elle applique la fonction de chiffrement $C_{11}(x) = x + 11$ à chacun des nombres : "**13 25 5 13 25 5**" ce qui correspond au mot crypté "**NZFNZF**". Elle transmet le mot crypté à Bruno, qui selon le même principe applique la fonction de déchiffrement $D_{11}(x) = x - 11$.



Exemple 18. Un exemple classique est le "rot13" (pour rotation par un décalage de 13) :

$$C_{13}(x) = x + 13$$

et comme $-13 \equiv 13 \pmod{26}$ alors $D_{13}(x) = x + 13$. La fonction de déchiffrement est la même que la fonction de chiffrement !
Exemple : déchiffrez le mot "**PRFNE**".

Notons ici deux points importants pour la suite : tout d'abord nous avons naturellement considéré un mot comme une succession de lettres, et chaque opération de chiffrement et déchiffrement s'effectue sur un bloc d'une seule lettre. Ensuite nous avons vu que chiffrer un message est une opération mathématique (certes sur un ensemble un peu spécial).

1.5 Espace des clés et attaque

Combien existe-t-il de possibilités de chiffrement par la méthode de César ? Il y a 26 fonctions C_k différentes, $k = 0, 1, \ldots, 25$. Encore une fois, k appartient à $\mathbb{Z}/26\mathbb{Z}$, car par exemple les fonctions C_{29} et C_3 sont identiques. Le décalage k s'appelle la ***clé de chiffrement***, c'est l'information nécessaire pour crypter le message. Il y a donc 26 clés différentes et l'***espace des clés*** est $\mathbb{Z}/26\mathbb{Z}$.

Il est clair que ce chiffrement de César est d'une sécurité très faible. Si Alice envoie un message secret à Bruno et que Chloé intercepte ce message, il sera facile pour Chloé de le décrypter même si elle ne connaît pas la clé secrète k. L'attaque la plus simple pour Chloé est de tester ce que donne chacune des 26 combinaisons possibles et de reconnaître parmi ces combinaisons laquelle donne un message compréhensible.

1.6 Algorithmes

Les ordinateurs ont révolutionné la cryptographie et surtout le décryptage d'un message intercepté. Nous montrons ici, à l'aide du langage `Python` comment programmer et attaquer le chiffrement de César. Tout d'abord la fonction de chiffrement se programme en une seule ligne :

cesar.py (1)

```
def cesar_chiffre_nb(x,k):
    return (x+k)%26
```

Ici `x` est un nombre de $\{0, 1, \ldots, 25\}$ et `k` est le décalage. `(x+k)%26` renvoie le reste modulo 26 de la somme `(x+k)`. Pour le décryptage, c'est aussi simple :

cesar.py (2)

```
def cesar_dechiffre_nb(x,k):
    return (x-k)%26
```

Pour chiffrer un mot ou un phrase, il n'y a pas de problèmes théoriques, mais seulement des difficultés techniques :

- Un mot ou une phrase est une chaîne de caractères, qui en fait se comporte comme une liste. Si `mot` est une chaîne alors `mot[0]` est la première lettre, `mot[1]` la deuxième lettre... et la boucle `for lettre in mot:` permet de parcourir chacune des lettres.
- Pour transformer une lettre en un nombre, on utilise le code Ascii qui à chaque caractère associe un nombre, `ord(A)` vaut 65, `ord(B)` vaut 66... Ainsi `(ord(lettre) - 65)` renvoie le rang de la lettre entre 0 et 25 comme nous l'avons fixé dès le départ.
- La transformation inverse se fait par la fonction `char` : `char(65)` renvoie le caractère `A`, `char(66)` renvoie `B`...
- Pour ajouter une lettre à une liste, faites `maliste.append(lettre)`. Enfin pour transformer une liste de caractères en une chaîne, faites `"".join(maliste)`.

Ce qui donne :

cesar.py (3)

```
def cesar_chiffre_mot(mot,k):
    mot_crypte = []                          # Liste vide
    for lettre in mot:                       # Pour chaque lettre
        nb = ord(lettre)-65                  # Lettre devient nb de 0 à 25
        nb_crypte = cesar_chiffre_nb(nb,k) # Chiffrement de César
        lettre_crypte = chr(nb_crypte+65)  # Retour aux lettres
        mot_crypte.append(lettre_crypte)   # Ajoute lettre au message
    mot_crypte = "".join(mot_crypte)         # Revient à chaine caractères
    return(mot_crypte)
```

Pour l'attaque on parcourt l'intégralité de l'espace des clés : k varie de 0 à 25. Noter que pour décrypter les messages on utilise ici simplement la fonction de César avec la clé $-k$.

cesar.py (4)

```
def cesar_attaque_mot(mot):
    for k in range(26):
        print "%s␣pour␣k=%d" % (cesar_chiffre_mot(mot,-k) , k)
    return None
```

2 Le chiffrement de Vigenère

2.1 Chiffrement mono-alphabétique

Principe

Nous avons vu que le chiffrement de César présente une sécurité très faible, la principale raison est que l'espace des clés est trop petit : il y a seulement 26 clés possibles, et on peut attaquer un message chiffré en testant toutes les clés à la main.
Au lieu de faire correspondre circulairement les lettres, on associe maintenant à chaque lettre une autre lettre (sans ordre fixe ou règle générale).
Par exemple :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	Q	B	M	X	I	T	E	P	A	L	W	H	S	D	O	Z	K	V	G	R	C	N	Y	J	U

Pour crypter le message

ETRE OU NE PAS ETRE TELLE EST LA QUESTION

on regarde la correspondance et on remplace la lettre **E** par la lettre **X**, puis la lettre **T** par la lettre **G**, puis la lettre **R** par la lettre **K**...
Le message crypté est alors :

XGKX DR SX OFV XGKX GXWWX XVG WF ZRXVGPDS

Pour le décrypter, en connaissant les substitutions, on fait l'opération inverse.

Avantage : nous allons voir que l'espace des clés est gigantesque et qu'il n'est plus question d'énumérer toutes les possibilités.
Inconvénients : la clé à retenir est beaucoup plus longue, puisqu'il faut partager la clé constituée des 26 lettres "FQBMX...". Mais surtout, nous allons voir que finalement ce protocole de chiffrement est assez simple à « craquer ».

Espace des clés

Mathématiquement, le choix d'une clé revient au choix d'une bijection de l'ensemble $\{A,B,\ldots,Z\}$ vers le même ensemble $\{A,B,\ldots,Z\}$. Il y a 26! choix possibles. En effet pour la lettre A de l'ensemble de départ, il y a 26 choix possibles (nous avions choisi F), pour B il reste 25 choix possibles (tout sauf F qui est déjà choisi), pour C il reste 24 choix... enfin pour Z il ne reste qu'une seule possibilité, la seule lettre non encore choisie. Au final il y a : $26 \times 25 \times 24 \times \cdots \times 2 \times 1$ soit 26! choix de clés. Ce qui fait environ 4×10^{26} clés. Il y a plus de clés différentes que de grains de sable sur Terre ! Si un ordinateur pouvait tester 1 000 000 de clés par seconde, il lui faudrait alors 12 millions d'années pour tout énumérer.

Attaque statistique

La principale faiblesse du chiffrement mono-alphabétique est qu'une même lettre est toujours chiffrée de la même façon. Par exemple, ici **E** devient **X**. Dans les textes longs, les lettres n'apparaissent pas avec la même fréquence. Ces fréquences varient suivant la langue utilisée. En français, les lettres les plus rencontrées sont dans l'ordre :

E S A I N T R U L O D C P M V Q G F H B X J Y Z K W

avec les fréquences (souvent proches et dépendant de l'échantillon utilisé) :

E	**S**	**A**	**I**	**N**	**T**	**R**	**U**	**L**	**O**	**D**
14.69%	8.01%	7.54%	7.18%	6.89%	6.88%	6.49%	6.12%	5.63%	5.29%	3.66%

Voici la méthode d'attaque : dans le texte crypté, on cherche la lettre qui apparaît le plus, et si le texte est assez long cela devrait être le chiffrement du **E**, la lettre qui apparaît ensuite dans l'étude des fréquences devrait être le chiffrement du **S**, puis le chiffrement du **A**... On obtient des morceaux de texte clair sous la forme d'une texte à trous et il faut ensuite deviner les lettres manquantes.

Par exemple, déchiffrons la phrase :

LHLZ HFQ BC HFFPZ WH YOUPFH MUPZH

On compte les apparitions des lettres :

H : 6 **F** : 4 **P** : 3 **Z** : 3

On suppose donc que le **H** crypte la lettre **E**, le **F** la lettre **S**, ce qui donne

***E** ES* ** ESS** *E ***SE ****E**

D'après les statistiques **P** et **Z** devraient se décrypter en **A** et **I** (ou **I** et **A**). Le quatrième mot "**HFFPZ**", pour l'instant décrypté en "**ESS****", se complète donc en "**ESSAI**" ou "**ESSIA**". La première solution semble correcte ! Ainsi **P** crypte **A**, et **Z** crypte **I**. La phrase est maintenant :

***E*I ES* ** ESSAI *E ***ASE **AIE**

En réfléchissant un petit peu, on décrypte le message :

CECI EST UN ESSAI DE PHRASE VRAIE

2.2 Le chiffrement de Vigenère

Blocs

L'espace des clés du chiffrement mono-alphabétique est immense, mais le fait qu'une lettre soit toujours cryptée de la même façon représente une trop grande faiblesse. Le chiffrement de Vigenère remédie à ce problème. On regroupe les lettres de notre texte par blocs, par exemple ici par blocs de longueur 4 :

CETTE PHRASE NE VEUT RIEN DIRE

devient

CETT EPHR ASEN EVEU TRIE NDIR E

(les espaces sont purement indicatifs, dans la première phrase ils séparent les mots, dans la seconde ils séparent les blocs).

Si k est la longueur d'un bloc, alors on choisit une clé constituée de k nombres de 0 à 25 : $(n_1,n_2,\ldots,n_k)$. Le chiffrement consiste à effectuer un chiffrement de César, dont le décalage dépend du rang de la lettre dans le bloc :

– un décalage de n_1 pour la première lettre de chaque bloc,

– un décalage de n_2 pour la deuxième lettre de chaque bloc,
– ...
– un décalage de n_k pour la k-ème et dernière lettre de chaque bloc.

Pour notre exemple, si on choisit comme clé $(3,1,5,2)$ alors pour le premier bloc "**CETT**" :
– un décalage de 3 pour **C** donne **F**,
– un décalage de 1 pour **E** donne **F**,
– un décalage de 5 pour le premier **T** donne **Y**,
– un décalage de 2 pour le deuxième **T** donne **V**.
Ainsi "**CETT**" de vient "**FFYV**". Vous remarquez que les deux lettres **T** ne sont pas cryptées par la même lettre et que les deux **F** ne cryptent pas la même lettre. On continue ensuite avec le deuxième bloc...

Mathématiques

L'élément de base n'est plus une lettre mais un ***bloc***, c'est-à-dire un regroupement de lettres. La fonction de chiffrement associe à un bloc de longueur k, un autre bloc de longueur k, ce qui donne en mathématisant les choses :

$$C_{n_1,n_2,\dots,n_k} : \left\{ \begin{array}{rcl} \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \times \cdots \times \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \times \cdots \times \mathbb{Z}/26\mathbb{Z} \\ (x_1, x_2, \dots, x_k) & \longmapsto & (x_1+n_1, x_2+n_2, \dots, x_k+n_k) \end{array} \right.$$

Chacune des composantes de cette fonction est un chiffrement de César. La fonction de déchiffrement est juste $C_{-n_1,-n_2,\dots,-n_k}$.

Espace des clés et attaque

Il y a 26^k choix possibles de clés, lorsque les blocs sont de longueur k. Pour des blocs de longueur $k = 4$ cela en donne déjà 456 976, et même si un ordinateur teste toutes les combinaisons possibles sans problème, il n'est pas question de parcourir cette liste pour trouver le message en clair, c'est-à-dire celui qui est compréhensible !
Il persiste tout de même une faiblesse du même ordre que celle rencontrée dans le chiffrement monoalphabétique : la lettre **A** n'est pas toujours cryptée par la même lettre, mais si deux lettres **A** sont situées à la même position dans deux blocs différents (comme par exemple "**ALPH ABET**") alors elles seront cryptées par la même lettre.
Une attaque possible est donc la suivante : on découpe notre message en plusieurs listes, les premières lettres de chaque bloc, les deuxièmes lettres de chaque bloc... et on fait une attaque statistique sur chacun de ces regroupements. Ce type d'attaque n'est possible que si la taille des blocs est petite devant la longueur du texte.

2.3 Algorithmes

Voici un petit algorithme qui calcule la fréquence de chaque lettre d'une phrase.

statistiques.py

```
def statistiques(phrase):
    liste_stat = [0 for x in range(26)]      # Une liste avec des 0
    for lettre in phrase:                    # On parcourt la phrase
        i = ord(lettre)-65
        if 0 <= i < 26:                      # Si c'est une vraie lettre
          liste_stat[i] = liste_stat[i] + 1
    return(liste_stat)
```

Et voici le chiffrement de Vigenère.

vigenere.py

```
def vigenere(mot,cle):                  # Clé est du type [n_1,...,n_k]
    mot_crypte = []
    k = len(cle)                        # Longueur de la clé
    i = 0                               # Rang dans le bloc
    for lettre in mot:                  # Pour chaque lettre
        nomb = ord(lettre)-65           # Lettre devient nb de 0 à 25
        nomb_code = (nomb+cle[i]) % 26  # Vigenère : on ajoute n_i
        lettre_code = chr(nomb_code+65) # On repasse aux lettres
        i=(i+1) % k                     # On passe au rang suivant
        mot_crypte.append(lettre_code)  # Ajoute lettre au message
    mot_crypte = "".join(mot_crypte)    # Revient à chaine caractères
    return(mot_crypte)
```

3 La machine Enigma et les clés secrètes

3.1 Un secret parfait

L'inconvénient des chiffrements précédents est qu'une même lettre est régulièrement chiffrée de la même façon, car la correspondance d'un alphabet à un ou plusieurs autres est fixée une fois pour toutes, ce qui fait qu'une attaque statistique est toujours possible. Nous allons voir qu'en changeant la correspondance à chaque lettre, il est possible de créer un chiffrement parfait !
Expliquons d'abord le principe à l'aide d'une analogie : j'ai choisi deux entiers m et c tels que $m+c=100$. Que vaut m ? C'est bien sûr impossible de répondre car il y a plusieurs possibilités : $0+100$, $1+99$, $2+98$,... Par contre, si je vous donne aussi c alors vous trouvez m immédiatement $m=100-c$.

Voici le principe du chiffrement : Alice veut envoyer à Bruno le message secret M suivant :

ATTAQUE LE CHATEAU

Alice a d'abord choisi une clé secrète C qu'elle a transmise à Bruno. Cette clé secrète est de la même longueur que le message (les espaces ne comptent pas) et composée d'entiers de 0 à 25, tirés au hasard. Par exemple C :

[4, 18, 2, 0, 21, 12, 18, 13, 7, 11, 23, 22, 19, 2, 16, 9]

Elle crypte la première lettre par un décalage de César donné par le premier entier : **A** est décalé de 4 lettres et devient donc **E**. La seconde lettre est décalée du second entier : le premier **T** devient **L**. Le second **T** est lui décalé de 2 lettres, il devient **V**. Le **A** suivant est décalé de 0 lettre, il reste **A**... Alice obtient un message chiffré X qu'elle transmet à Bruno :

ELVALGW YL NEWMGQD

Pour le décrypter, Bruno, qui connaît la clé, n'a qu'à faire le décalage dans l'autre sens.
Notez que deux lettres identiques (par exemples les **T**) n'ont aucune raison d'être cryptées de la même façon. Par exemple, les **T** du message initial sont cryptés dans l'ordre par un **L**, un **V** et un **M**.

Formalisons un peu cette opération. On identifie A avec 0, B avec 1, ..., Z avec 25. Alors le message crypté X est juste la "somme" du message M avec la clé secrète C, la somme s'effectuant lettre à lettre, terme à terme, modulo 26.
Notons cette opération $M \oplus C = X$.

	A	T	T	A	Q	U	E	L	E	C	H	A	T	E	A	U
	0	19	19	0	16	20	4	11	4	2	7	0	19	4	0	20
$\oplus$																
	4	18	2	0	21	12	18	13	7	11	23	22	19	2	16	9
=	4	11	21	0	11	6	22	24	11	13	4	22	12	6	16	3
	E	L	V	A	L	G	W	Y	L	N	E	W	M	G	Q	D

Bruno reçoit X et connaît C, il effectue donc $X \ominus C = M$.
Pourquoi ce système est-il inviolable ? Pour chacune des lettres, c'est exactement le même problème que trouver m, sachant que $m + c = x$ (où $x = 100$), mais sans connaître c. Toutes les possibilités pour m pourraient être juste. Et bien sûr, dès que l'on connaît c, la solution est triviale : $m = x - c$.
Il y a trois principes à respecter pour que ce système reste inviolable :

1. La longueur de la clé est égale à la longueur du message.
2. La clé est choisie au hasard.
3. La clé ne sert qu'une seule fois.

Ce système appelé "masque jetable" ou chiffrement de Vernam est parfait en théorie, mais sa mise en œuvre n'est pas pratique du tout ! Tout d'abord il faut que la clé soit aussi longue que le message. Pour un message court cela ne pose pas de problème, mais pour envoyer une image par exemple cela devient très lourd. Ensuite, il faut trouver un moyen sûr d'envoyer la clé secrète à son interlocuteur avant de lui faire parvenir le message. Et il faut recommencer cette opération à chaque message, ou bien se mettre d'accord dès le départ sur un ***carnet de clés*** : une longue liste de clés secrètes.
Pour justifier que ce système est vraiment inviolable voici une expérience amusante : Alice veut envoyer le message M ="**ATTAQUE LE CHATEAU**" à Bruno, elle choisit la clé secrète C=[4, 18, 2, 0,...] comme ci-dessus et obtient le message chiffré X="**ELVA**..." qu'elle transmet à Bruno.
Alice se fait kidnapper par Chloé, qui veut l'obliger à déchiffrer son message. Heureusement, Alice a anticipé les soucis : elle a détruit le message M, la clé secrète C et a créé un faux message M' et une fausse clé secrète C'. Alice fournit cette fausse clé secrète C' à Chloé, qui déchiffre le message par l'opération $X \ominus C'$ et elle trouve le message bien inoffensif M' :

RECETTE DE CUISINE

Alice est innocentée !
Comment est-ce possible ? Alice avait au préalable préparé un message neutre M' de même longueur que M et calculé la fausse clé secrète $C' = X \ominus M'$. Chloé a obtenu (par la contrainte) X et C', elle déchiffre le message ainsi

$$X \ominus C' = X \ominus (X \ominus M') = (X \ominus X) \oplus M' = M'$$

Chloé trouve donc le faux message.
Ici la fausse clé C' est :

[13, 7, 19, 22, 18, 13, 18, 21, 7, 11, 10, 14, 20, 24, 3, 25]

La première lettre du message chiffré est un **E**, en reculant de 13 lettres dans l'alphabet, elle se déchiffre en **R**...

3.2 La machine Enigma

Afin de s'approcher de ce protocole de chiffrement parfait, il faut trouver un moyen de générer facilement de longues clés, comme si elles avaient été générées au hasard. Nous allons étudier deux exemples utilisés en pratique à la fin du siècle dernier, une méthode électro-mécanique : la machine Enigma et une méthode numérique : le DES.
La machine Enigma est une machine électro-mécanique qui ressemble à une machine à écrire. Lorsque qu'une touche est enfoncée, des disques internes sont actionnés et le caractère crypté s'allume. Cette machine, qui sert aussi au déchiffrement, était utilisée pour les communications de l'armée allemande durant la seconde guerre mondiale. Ce que les Allemands ne savaient pas, c'est que les services secrets polonais et britanniques avaient réussi à percer les secrets de cette machine et étaient capables de

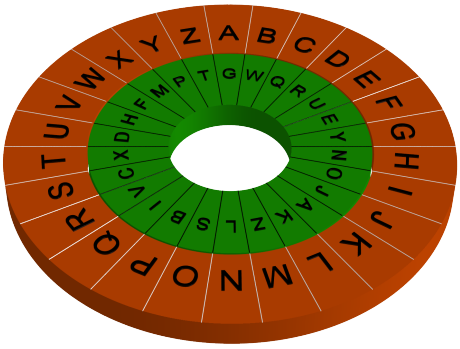
déchiffrer les messages transmis par les allemands. Ce long travail d'études et de recherches a nécessité tout le génie d'Alan Turing et l'invention de l'ancêtre de l'ordinateur.

Nous symbolisons l'élément de base de la machine Enigma par deux anneaux :

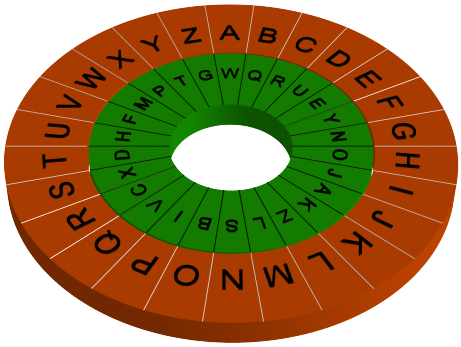
- Un anneau extérieur contenant l'alphabet "**ABCDE**..." symbolisant le clavier de saisie des messages. Cet anneau est fixe.
- Un anneau intérieur contenant un alphabet dans le désordre (sur la figure "**GWQRU**..."). Cet anneau est mobile et effectue une rotation à chaque touche tapée au clavier. Il représente la clé secrète.

Voici, dans ce cas, le processus de chiffrement du mot "**BAC**", avec la clé de chiffrement "**G**" :

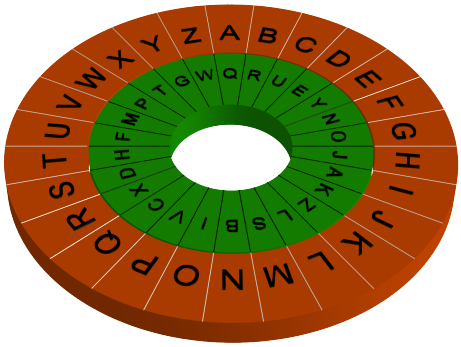
1. **Position initiale.** L'opérateur tourne l'anneau intérieur de sorte que le **A** extérieur et fixe soit en face du **G** intérieur (et donc **B** en face de **W**).



2. **Première lettre.** L'opérateur tape la première lettre du message : **B**, la machine affiche la correspondance **W**.
3. **Rotation.** L'anneau intérieur tourne de 1/26ème de tour, maintenant le **A** extérieur et fixe est en face du **W**, le **B** en face du **Q**,...

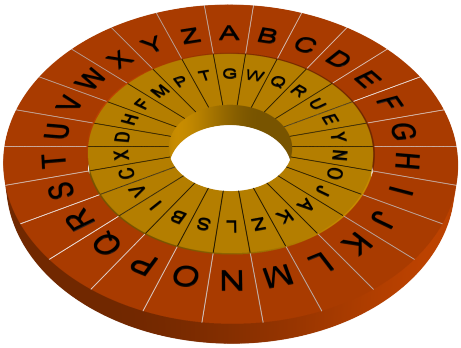


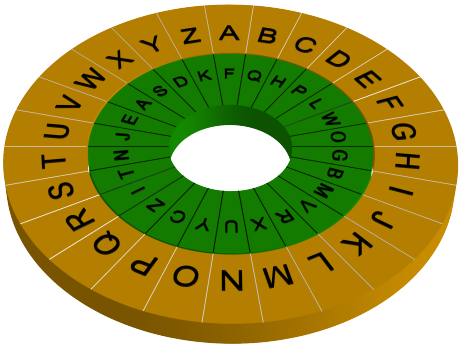
4. **Deuxième lettre.** L'opérateur tape la deuxième lettre du message **A**, la machine affiche la correspondance, c'est de nouveau **W**.
5. **Rotation.** L'anneau intérieur tourne de 1/26ème de tour, maintenant le **A** extérieur et fixe est en face du **Q**, le **B** en face du **R**, le **C** en face du **U**,...



6. **Troisième lettre.** L'opérateur tape la troisième lettre du message **C**, la machine affiche la correspondance **U**.
7. **Rotation.** L'anneau intérieur effectue sa rotation.
8. **Message chiffré.** Le message crypté est donc "**WWU**"

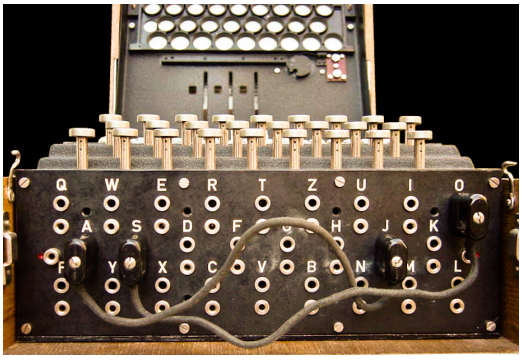
Cette méthode de chiffrement est identique à un chiffrement de type Vigenère pour une clé de longueur 26. Il y a 26 clés différents à disposition avec un seul anneau intérieur et identifiées par lettre de la position initiale : **G**, **W**, **Q**... **T** correspondant aux alphabets : "**GWQ...PT**", "**WQR...TG**", "**QRU...GW**"... En fait, la machine Enigma était beaucoup plus sophistiquée, il n'y avait pas un mais plusieurs anneaux intérieurs. Par exemple pour deux anneaux intérieurs comme sur la figure : **B** s'envoie sur **W**, qui s'envoie sur **A**; la lettre **B** est cryptée en **A**. Ensuite l'anneau intérieur numéro 1 effectue 1/26ème de tour. La lettre **A** s'envoie sur **W**, qui s'envoie sur **A**; la lettre **A** est cryptée en **A**. Lorsque l'anneau intérieur numéro 1 a fait une rotation complète (26 lettres ont été tapées) alors l'anneau intérieur numéro 2 effectue 1/26ème de tour. C'est comme sur un compteur kilométrique, lorsque le chiffre des kilomètres parcourt $0,1,2,3,...,9$, alors au kilomètre suivant, le chiffre des kilomètres est 0 et celui des dizaines de kilomètres est augmenté d'une unité.





S'il y a trois anneaux, lorsque l'anneau intérieur 2 a fait une rotation complète, l'anneau intérieur 3 tourne de 1/26ème de tour. Il y a alors 26^3 clés différentes facilement identifiables par les trois lettres des positions initiales des anneaux.

Il fallait donc pour utiliser cette machine, d'abord choisir les disques (nos anneaux intérieurs) les placer dans un certain ordre, fixer la position initiale de chaque disque. Ce système était rendu largement plus complexe avec l'ajout de correspondances par fichage entre les lettres du clavier (voir photo). Le nombre de clés possibles dépassait plusieurs milliards de milliards !



3.3 La ronde des chiffres : DES

La machine Enigma génère mécaniquement un alphabet différent à chaque caractère crypté, tentant de se rapprocher d'un chiffrement parfait. Nous allons voir une autre méthode, cette fois numérique : le DES. Le DES (*Data Encryption Standard*) est un protocole de chiffrement par blocs. Il a été, entre 1977 et 2001, le standard de chiffrement pour les organisations du gouvernement des États-Unis et par extension pour un grand nombre de pays dans le monde.

Commençons par rappeler que l'objectif est de générer une clé aléatoire de grande longueur. Pour ne pas avoir à retenir l'intégralité de cette longue clé, on va la générer de façon pseudo-aléatoire à partir d'une petite clé.
Voyons un exemple élémentaire de suite pseudo-aléatoire.
Soit (u_n) la suite définie par la donnée de (a,b) et de u_0 et la relation de récurrence

$$u_{n+1} \equiv a \times u_n + b \pmod{26}.$$

Par exemple pour $a=2$, $b=5$ et $u_0=6$, alors les premiers termes de la suites sont :

$$6 \quad 17 \quad 13 \quad 5 \quad 15 \quad 9 \quad 23 \quad 25 \quad 3 \quad 11 \quad 1 \quad 7 \quad 19 \quad 17 \quad 13 \quad 5$$

Les trois nombres (a,b,u_0) représentent la clé principale et la suite des $(u_n)_{n\in\mathbb{N}}$ les clés secondaires.
Avantages : à partir d'une clé principale on a généré une longue liste de clés secondaires. Inconvénients : la liste n'est pas si aléatoire que cela, elle se répète ici avec une période de longueur 12 : $17,13,5,\ldots,17,13,5,\ldots$

Le système DES est une version sophistiquée de ce processus : à partir d'une clé courte et d'opérations élémentaires on crypte un message. Comme lors de l'étude de la machine Enigma, nous allons présenter une version très simplifiée de ce protocole afin d'en expliquer les étapes élémentaires.
Pour changer, nous allons travailler modulo 10. Lorsque l'on travaille par blocs, les additions se font *bit* par *bit*. Par exemple : $[1\ 2\ 3\ 4] \oplus [7\ 8\ 9\ 0] = [8\ 0\ 2\ 4]$ car $(1+7 \equiv 8 \pmod{10}, 2+8 \equiv 0 \pmod{10}, \ldots)$
Notre message est coupé en blocs, pour nos explications ce seront des blocs de longueur 8. La clé est de longueur 4.
Voici le message (un seul bloc) : $M = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$ et voici la clé : $C = [3\ 1\ 3\ 2]$.

Étape 0. Initialisation. On note $M_0 = M$ et on découpe M en une partie gauche et une partie droite

$$M_0 = [G_0 \parallel D_0] = [1\ 2\ 3\ 4 \parallel 5\ 6\ 7\ 8]$$

Étape 1. Premier tour. On pose

$$M_1 = [D_0 \parallel C \oplus \sigma(G_0)]$$

où σ est une permutation circulaire.
On effectue donc trois opérations pour passer de M_0 à M_1 :

1. On échange la partie droite et la partie gauche de M_0 :

$$M_0 \longmapsto [5\ 6\ 7\ 8 \parallel 1\ 2\ 3\ 4]$$

2. Sur la nouvelle partie droite, on permute circulairement les nombres :

$$\longmapsto [5\ 6\ 7\ 8 \parallel 2\ 3\ 4\ 1]$$

3. Puis on ajoute la clé secrète C à droite (ici $C = [3\ 1\ 3\ 2]$) :

$$\longmapsto [5\ 6\ 7\ 8 \parallel 5\ 4\ 7\ 3] = M_1$$

On va recommencer le même processus. Cela revient à appliquer la formule de récurrence, qui partant de $M_i = [G_i \parallel D_i]$, définit

$$M_{i+1} = [D_i \parallel C \oplus \sigma(G_i)]$$

Étape 2. Deuxième tour. On part de $M_1 = [5\ 6\ 7\ 8 \parallel 5\ 4\ 7\ 3]$.

1. On échange la partie droite et la partie gauche de M_0 :

$$M_0 \longmapsto [5\ 4\ 7\ 3 \parallel 5\ 6\ 7\ 8]$$

2. Sur la nouvelle partie droite, on permute circulairement les nombres.

$$\longmapsto [5\ 4\ 7\ 3 \parallel 6\ 7\ 8\ 5]$$

3. Puis on ajoute la clé secrète C à droite.

$$\longmapsto [5\ 4\ 7\ 3 \parallel 9\ 8\ 1\ 7] = M_2$$

On peut décider de s'arrêter après ce tour et renvoyer le message crypté $X = M_2 = [5\ 4\ 7\ 3\ 9\ 8\ 1\ 7]$. Comme chaque opération élémentaire est inversible, on applique un protocole inverse pour déchiffrer. Dans le vrai protocole du DES, la clé principale est de taille 64 *bits*, il y a plus de manipulations sur le message et les étapes mentionnées ci-dessus sont effectuées 16 fois (on parle de tours). À chaque tour, une clé différente est utilisée. Il existe donc un préambule à ce protocole : générer 16 clés secondaires (de longueur 48 *bits*) à partir de la clé principale, ce qui se fait selon le principe de la suite pseudo-aléatoire (u_n) expliquée plus haut.

4 La cryptographie à clé publique

Les Grecs pour envoyer des messages secrets rasaient la tête du messager, tatouaient le message sur son crâne et attendaient que les cheveux repoussent avant d'envoyer le messager effectuer sa mission ! Il est clair que ce principe repose uniquement sur le secret de la méthode.

4.1 Le principe de Kerckhoffs

Cette méthode rudimentaire va à l'encontre du principe de Kerckhoffs. Le principe de Kerckhoffs s'énonce ainsi :

«La sécurité d'un système de chiffrement ne doit reposer que sur la clé.»

Cela se résume aussi par :

«L'ennemi peut avoir connaissance du système de chiffrement.»

Voici le texte original d'Auguste Kerckhoffs de 1883 «La cryptographie militaire» paru dans le *Journal des sciences militaires*.
Il traite notamment des enjeux de sécurité lors des correspondances :

> *«Il faut distinguer entre un système d'écriture chiffré, imaginé pour un échange momentané de lettres entre quelques personnes isolées, et une méthode de cryptographie destinée à régler pour un temps illimité la correspondance des différents chefs d'armée entre eux.»*

Le principe fondamental est le suivant :

> *«Dans le second cas, [...] il faut que* **le système n'exige pas le secret**, *et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.»*

Ce principe est novateur dans la mesure où intuitivement il semble opportun de dissimuler le maximum de choses possibles : clé et système de chiffrement utilisés. Mais l'objectif visé par Kerckhoffs est plus académique, il pense qu'un système dépendant d'un secret mais dont le mécanisme est connu de tous sera testé, attaqué, étudié, et finalement utilisé s'il s'avère intéressant et robuste.

4.2 Factorisations des entiers

Quels outils mathématiques répondent au principe de Kerckoffs ?
Un premier exemple est la toute simple multiplication ! En effet si je vous demande combien font 5×7, vous répondez 35. Si je vous demande de factoriser 35 vous répondez 5×7. Cependant ces deux questions ne sont pas du même ordre de difficulté. Si je vous demande de factoriser 1591, vous aller devoir faire plusieurs tentatives, alors que si je vous avais directement demandé de calculer 37×43 cela ne pose pas de problème.
Pour des entiers de plusieurs centaines de chiffres le problème de factorisation ne peut être résolu en un temps raisonnable, même pour un ordinateur. C'est ce problème asymétrique qui est à la base de la cryptographie RSA (que nous détaillerons plus tard) : connaître p et q apporte plus d'information utilisable que $p \times q$. Même si en théorie à partir de $p \times q$ on peut retrouver p et q, en pratique ce sera impossible.

Formalisons ceci avec la notion de complexité. La ***complexité*** est le temps de calculs (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération.
Commençons par la complexité de l'addition : disons que calculer la somme de deux chiffres (par exemple $6+8$) soit de complexité 1 (par exemple 1 seconde pour un humain, 1 milliseconde pour un ordinateur). Pour calculer la somme de deux entiers à n chiffres, la complexité est d'ordre n (exemple : $1234+2323$, il faut faire 4 additions de chiffres, donc environ 4 secondes pour un humain).
La multiplication de deux entiers à n chiffres est de complexité d'ordre n^2. Par exemple pour multiplier 1234 par 2323 il faut faire 16 multiplications de chiffres (chaque chiffre de 1234 est à multiplier par chaque chiffre de 2323).
Par contre la meilleure méthode de factorisation connue est de complexité d'ordre $\exp(4n^{\frac{1}{3}})$ (c'est moins que $\exp(n)$, mais plus que n^d pour tout d, lorsque n tend vers $+\infty$).
Voici un tableau pour avoir une idée de la difficulté croissante pour multiplier et factoriser des nombres à n chiffres :

n	multiplication	factorisation
3	9	320
4	16	572
5	25	934
10	100	5 528
50	2 500	2 510 835
100	10 000	115 681 968
200	40 000	14 423 748 780

4.3 Fonctions à sens unique

Il existe bien d'autres situations mathématiques asymétriques : les ***fonctions à sens unique***. En d'autres termes, étant donnée une fonction f, il est possible connaissant x de calculer «facilement» $f(x)$; mais connaissant un élément de l'ensemble image de f, il est «difficile» ou impossible de trouver son antécédent.

Dans le cadre de la cryptographie, posséder une fonction à sens unique qui joue le rôle de chiffrement n'a que peu de sens. En effet, il est indispensable de trouver un moyen efficace afin de pouvoir déchiffrer les messages chiffrés. On parle alors de ***fonction à sens unique avec trappe secrète***.

Prenons par exemple le cas de la fonction f suivante :

$$f : x \longmapsto x^3 \pmod{100}.$$

- Connaissant x, trouver $y = f(x)$ est facile, cela nécessite deux multiplications et deux divisions.
- Connaissant y image par f d'un élément x ($y = f(x)$), retrouver x est difficile.

Tentons de résoudre le problème suivant : trouver x tel que $x^3 \equiv 11 \pmod{100}$.
On peut pour cela :
- soit faire une recherche exhaustive, c'est-à-dire essayer successivement 1, 2, 3, ..., 99, on trouve alors :

$$71^3 = 357\,911 \equiv 11 \pmod{100},$$

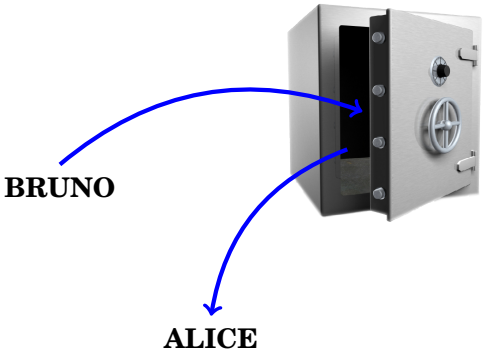
- soit utiliser la trappe secrète : $y \longmapsto y^7 \pmod{100}$ qui fournit directement le résultat !

$$11^7 = 19\,487\,171 \equiv 71 \pmod{100}.$$

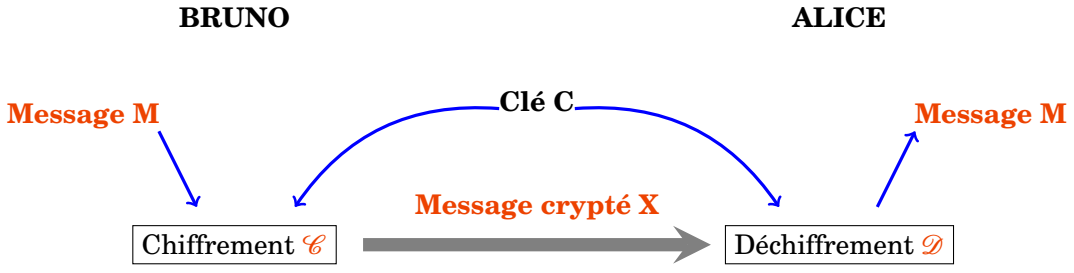
La morale est la suivante : le problème est dur à résoudre, sauf pour ceux qui connaissent la trappe secrète. (Attention, dans le cas de cet exemple, la fonction f n'est pas bijective.)

4.4 Chiffrement à clé privée

Petit retour en arrière. Les protocoles étudiés dans les chapitres précédents étaient des ***chiffrements à clé privée***. De façon imagée, tout se passe comme si Bruno pouvaient déposer son message dans un coffre fort pour Alice, Alice et Bruno étant les seuls à posséder la clé du coffre.

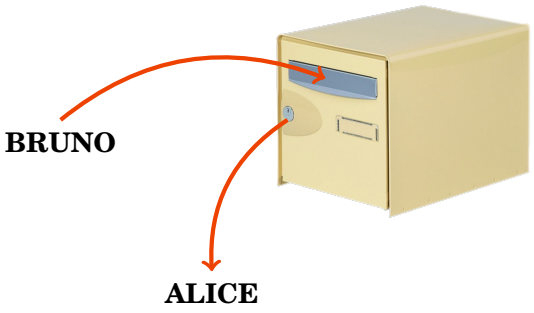


En effet, jusqu'ici, les deux interlocuteurs se partageaient une même clé qui servait à chiffrer (et déchiffrer) les messages. Cela pose bien sûr un problème majeur : Alice et Bruno doivent d'abord se communiquer la clé.

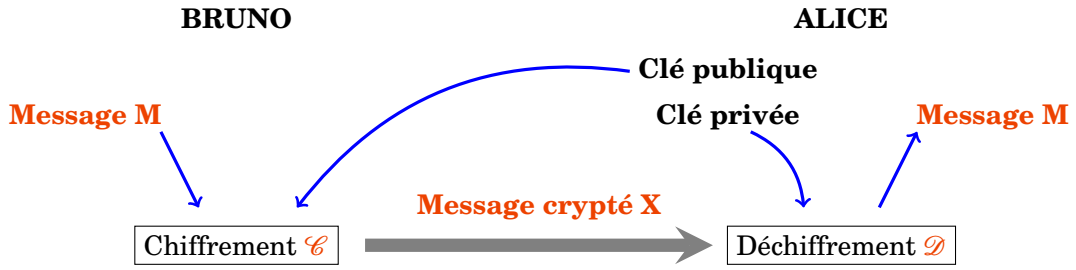


4.5 Chiffrement à clé publique

Les fonctions à sens unique à trappe donnent naissance à des protocoles de chiffrement à clé publique. L'association «clé» et «publique» peut paraître incongrue, mais il signifie que le principe de chiffrement est accessible à tous mais que le déchiffrement nécessite une clé qu'il faut bien sûr garder secrète.



De façon imagée, si Bruno veut envoyer un message à Alice, il dépose son message dans la boîte aux lettres d'Alice, seule Alice pourra ouvrir sa boîte et consulter le message. Ici la clé publique est symbolisée par la boîte aux lettre, tout le monde peut y déposer un message, la clé qui ouvre la boîte aux lettres est la clé privée d'Alice, que Alice doit conserver à l'abri.



En prenant appui sur l'exemple précédent, si le message initial est 71 et que la fonction f de chiffrement est connue de tous, le message transmis est 11 et le déchiffrement sera rapide si la trappe secrète 7 est connue du destinataire.
Les paramètres d'un protocole de ***chiffrement à clé publique*** sont donc :
– les fonctions de chiffrement et de déchiffrement : $\mathscr{C}$ et $\mathscr{D}$,
– la clé publique du destinataire qui va permettre de paramétrer la fonction $\mathscr{C}$,
– la clé privée du destinataire qui va permettre de paramétrer la fonction $\mathscr{D}$.
Dans le cadre de notre exemple Bruno souhaite envoyer un message à Alice, ces éléments sont :
– $\mathscr{C} : x \longmapsto x^? \pmod{100}$ et $\mathscr{D} : x \longmapsto x^? \pmod{100}$,
– **3** : la clé publique d'Alice qui permet de définir complètement la fonction de chiffrement :

$$\mathscr{C} : x \longmapsto x^3 \pmod{100},$$

– **7** : la clé privée d'Alice qui permet de définir complètement la fonction de déchiffrement :

$$\mathscr{D} : x \longmapsto x^7 \pmod{100}.$$

Dans la pratique, un chiffrement à clé publique nécessite plus de calculs et est donc assez lent, plus lent qu'un chiffrement à clé privée. Afin de gagner en rapidité, un protocole hybride peut être mis en place de la façon suivante :
– à l'aide d'un protocole de chiffrement à clé publique, Alice et Bruno échangent une clé,
– Alice et Bruno utilise cette clé dans un protocole de chiffrement à clé privée.

5 L'arithmétique pour RSA

Pour un entier n, sachant qu'il est le produit de deux nombres premiers, il est difficile de retrouver les facteurs p et q tels que $n = pq$. Le principe du chiffrement RSA, chiffrement à clé publique, repose sur cette difficulté.
Dans cette partie nous mettons en place les outils mathématiques nécessaires pour le calcul des clés publique et privée ainsi que les procédés de chiffrement et déchiffrement RSA.

5.1 Le petit théorème de Fermat amélioré

Nous connaissons le petit théorème de Fermat

Théorème 5 (Petit théorème de Fermat).
Si p est un nombre premier et $a \in \mathbb{Z}$ alors

$$\boxed{a^p \equiv a \pmod{p}}$$

et sa variante :

Corollaire 5. Si p ne divise pas a alors

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

Nous allons voir une version améliorée de ce théorème dans le cas qui nous intéresse :

Théorème 6 (Petit théorème de Fermat amélioré).
Soient p et q deux nombres premiers distincts et soit $n = pq$. Pour tout $a \in \mathbb{Z}$ tel que $\text{pgcd}(a,n) = 1$ alors :

$$\boxed{a^{(p-1)(q-1)} \equiv 1 \pmod{n}}$$

On note $\varphi(n) = (p-1)(q-1)$, la ***fonction d'Euler***. L'hypothèse $\text{pgcd}(a,n) = 1$ équivaut ici à ce que a ne soit divisible ni par p, ni par q. Par exemple pour $p = 5$, $q = 7$, $n = 35$ et $\varphi(n) = 4 \cdot 6 = 20$. Alors pour $a = 1,2,3,4,6,8,9,11,12,13,16,17,18,\ldots$ on a bien $a^{20} \equiv 1 \pmod{35}$.

Démonstration. Notons $c = a^{(p-1)(q-1)}$. Calculons c modulo p :

$$c \equiv a^{(p-1)(q-1)} \equiv (a^{(p-1)})^{q-1} \equiv 1^{q-1} \equiv 1 \pmod{p}$$

où l'on appliquer le petit théorème de Fermat : $a^{p-1} \equiv 1 \pmod{p}$, car p ne divise pas a.
Calculons ce même c mais cette fois modulo q :

$$c \equiv a^{(p-1)(q-1)} \equiv (a^{(q-1)})^{p-1} \equiv 1^{p-1} \equiv 1 \pmod{q}$$

où l'on appliquer le petit théorème de Fermat : $a^{q-1} \equiv 1 \pmod{q}$, car q ne divise pas a.
Conclusion partielle : $c \equiv 1 \pmod{p}$ et $c \equiv 1 \pmod{q}$.

Nous allons en déduire que $c \equiv 1 \pmod{pq}$.
Comme $c \equiv 1 \pmod{p}$ alors il existe $\alpha \in \mathbb{Z}$ tel que $c = 1 + \alpha p$; comme $c \equiv 1 \pmod{q}$ alors il existe $\beta \in \mathbb{Z}$ tel que $c = 1 + \beta q$. Donc $c - 1 = \alpha p = \beta q$. De l'égalité $\alpha p = \beta q$, on tire que $p | \beta q$.
Comme p et q sont premiers entre eux (car ce sont des nombres premiers distincts) alors par le lemme de Gauss on en déduit que $p | \beta$. Il existe donc $\beta' \in \mathbb{Z}$ tel que $\beta = \beta' p$.
Ainsi $c = 1 + \beta q = 1 + \beta' pq$. Ce qui fait que $c \equiv 1 \pmod{pq}$, c'est exactement dire $a^{(p-1)(q-1)} \equiv 1 \pmod{n}$.
□

5.2 L'algorithme d'Euclide étendu

Nous avons déjà étudié l'algorithme d'Euclide qui repose sur le principe que $\text{pgcd}(a,b) = \text{pgcd}(b, a \pmod b)$.
Voici sa mise en œuvre informatique.

euclide.py (1)

```
def euclide(a,b):
    while b !=0 :
        a , b = b , a % b
    return a
```

On profite que `Python` assure les affectations simultanées, ce qui pour nous correspond aux suites

$$\begin{cases} a_{i+1} & = & b_i \\ b_{i+1} & \equiv & a_i \pmod{b_i} \end{cases}$$

initialisée par $a_0 = a$, $b_0 = b$.
Nous avons vu aussi comment « remonter » l'algorithme d'Euclide à la main pour obtenir les coefficients de Bézout u, v tels que $au + bv = \text{pgcd}(a,b)$. Cependant il nous faut une méthode plus automatique pour obtenir ces coefficients, c'est l'***algorithme d'Euclide étendu***.
On définit deux suites (x_i), (y_i) qui vont aboutir aux coefficients de Bézout.
L'initialisation est :

$$x_0 = 1 \qquad x_1 = 0 \qquad y_0 = 0 \qquad y_1 = 1$$

et la formule de récurrence pour $i \geqslant 1$:

$$x_{i+1} = x_{i-1} - q_i x_i \qquad y_{i+1} = y_{i-1} - q_i y_i$$

où q_i est le quotient de la division euclidienne de a_i par b_i.

euclide.py (2)

```
def euclide_etendu(a,b):
    x = 1 ; xx = 0
    y = 0 ; yy = 1
    while b !=0 :
        q = a // b
        a , b = b , a % b
        xx , x = x - q*xx , xx
        yy , y = y - q*yy , yy
    return (a,x,y)
```

Cet algorithme renvoie d'abord le pgcd, puis les coefficients u, v tels que $au + bv = \text{pgcd}(a,b)$.

5.3 Inverse modulo n

Soit $a \in \mathbb{Z}$, on dit que $x \in \mathbb{Z}$ est un ***inverse de a modulo n*** si $\boxed{ax \equiv 1 \pmod n.}$
Trouver un inverse de a modulo n est donc un cas particulier de l'équation $ax \equiv b \pmod n$.

Proposition 8.– a admet un inverse modulo n si et seulement si a et n sont premiers entre eux.
– Si $au + nv = 1$ alors u est un inverse de a modulo n.

En d'autres termes, trouver un inverse de a modulo n revient à calculer les coefficients de Bézout associés à la paire (a,n).

Démonstration. La preuve est essentiellement une reformulation du théorème de Bézout :

$$\begin{aligned} \text{pgcd}(a,n)=1 &\iff \exists u,v\in\mathbb{Z} \quad au+nv=1 \\ &\iff \exists u\in\mathbb{Z} \quad au\equiv 1 \pmod{n} \end{aligned}$$

□

Voici le code :

euclide.py (3)

```
def inverse(a,n):
    c,u,v = euclide_etendu(a,n)    # pgcd et coeff. de Bézout
    if c != 1 :             # Si pgcd différent de 1 renvoie 0
        return 0
    else :
        return u % n                        # Renvoie l'inverse
```

5.4 L'exponentiation rapide

Nous aurons besoin de calculer rapidement des puissances modulo n. Pour cela il existe une méthode beaucoup plus efficace que de calculer d'abord a^k puis de le réduire modulo n. Il faut garder à l'esprit que les entiers que l'on va manipuler ont des dizaines voir des centaines de chiffres.
Voyons la technique sur l'exemple de $5^{11} \pmod{14}$. L'idée est de seulement calculer $5, 5^2, 5^4, 5^8 \ldots$ et de réduire modulo n à chaque fois. Pour cela on remarque que $11 = 8+2+1$ donc

$$5^{11} = 5^8 \times 5^2 \times 5^1.$$

Calculons donc les $5^{2^i} \pmod{14}$:

$$\begin{aligned} 5 &\equiv 5 \pmod{14} \\ 5^2 &\equiv 25 \equiv 11 \pmod{14} \\ 5^4 &\equiv 5^2 \times 5^2 \equiv 11 \times 11 \equiv 121 \equiv 9 \pmod{14} \\ 5^8 &\equiv 5^4 \times 5^4 \equiv 9 \times 9 \equiv 81 \equiv 11 \pmod{14} \end{aligned}$$

à chaque étape est effectuée une multiplication modulaire. Conséquence :

$$5^{11} \equiv 5^8 \times 5^2 \times 5^1 \equiv 11 \times 11 \times 5 \equiv 11 \times 55 \equiv 11 \times 13 \equiv 143 \equiv 3 \pmod{14}.$$

Nous obtenons donc un calcul de $5^{11} \pmod{14}$ en 5 opérations au lieu de 10 si on avait fait $5 \times 5 \times 5 \cdots$.
Voici une formulation générale de la méthode. On écrit le développement de l'exposant k en base 2 : $(k_\ell, \ldots, k_2, k_1, k_0)$ avec $k_i \in \{0,1\}$ de sorte que

$$k = \sum_{i=0}^{\ell} k_i 2^i.$$

On obtient alors

$$x^k = x^{\sum_{i=0}^{\ell} k_i 2^i} = \prod_{i=0}^{\ell} (x^{2^i})^{k_i}.$$

Par exemple 11 en base 2 s'écrit $(1,0,1,1)$, donc, comme on l'a vu :

$$5^{11} = (5^{2^3})^1 \times (5^{2^2})^0 \times (5^{2^1})^1 \times (5^{2^0})^1.$$

Voici un autre exemple : calculons $17^{154} \pmod{100}$. Tout d'abord on décompose l'exposant $k = 154$ en base 2 : $154 = 128+16+8+2 = 2^7+2^4+2^3+2^1$, il s'écrit donc en base 2 : $(1,0,0,1,1,0,1,0)$.

Ensuite on calcule $17, 17^2, 17^4, 17^8, ..., 17^{128}$ modulo 100.

$$
\begin{aligned}
17 &\equiv 17 \pmod{100} \\
17^2 &\equiv 17 \times 17 \equiv 289 \equiv 89 \pmod{100} \\
17^4 &\equiv 17^2 \times 17^2 \equiv 89 \times 89 \equiv 7921 \equiv 21 \pmod{100} \\
17^8 &\equiv 17^4 \times 17^4 \equiv 21 \times 21 \equiv 441 \equiv 41 \pmod{100} \\
17^{16} &\equiv 17^8 \times 17^8 \equiv 41 \times 41 \equiv 1681 \equiv 81 \pmod{100} \\
17^{32} &\equiv 17^{16} \times 17^{16} \equiv 81 \times 81 \equiv 6561 \equiv 61 \pmod{100} \\
17^{64} &\equiv 17^{32} \times 17^{32} \equiv 61 \times 61 \equiv 3721 \equiv 21 \pmod{100} \\
17^{128} &\equiv 17^{64} \times 17^{64} \equiv 21 \times 21 \equiv 441 \equiv 41 \pmod{100}
\end{aligned}
$$

Il ne reste qu'à rassembler :

$$17^{154} \equiv 17^{128} \times 17^{16} \times 17^8 \times 17^2 \equiv 41 \times 81 \times 41 \times 89 \equiv 3321 \times 3649 \equiv 21 \times 49 \equiv 1029 \equiv 29 \pmod{100}$$

On en déduit un algorithme pour le calcul rapide des puissances.

puissance.py

```
def puissance(x,k,n):
    puiss = 1                       # Initialisation du résultat
    while (k>0):
        if k % 2 != 0 :             # Si k est impair (i.e. k_i=1)
            puiss = (puiss*x) % n
        x = x*x % n                 # Vaut x, x^2, x^4,...
        k = k // 2
    return(puiss)
```

En fait `Python` sait faire l'exponentiation rapide : `pow(x,k,n)` pour le calcul de a^k modulo n, il faut donc éviter `(x ** k) % n` qui n'est pas adapté.

6 Le chiffrement RSA

Voici le but ultime de ce cours : la chiffrement RSA. Il est temps de relire l'introduction du chapitre « Arithmétique » pour s'apercevoir que nous sommes prêts !

> *Pour crypter un message on commence par le transformer en un –ou plusieurs– nombres. Les processus de chiffrement et déchiffrement font appel à plusieurs notions :*
> - *On choisit deux* **nombres premiers** *p et q que l'on garde secrets et on pose $n = p \times q$. Le principe étant que même connaissant n il est très difficile de retrouver p et q (qui sont des nombres ayant des centaines de chiffres).*
> - *La clé secrète et la clé publique se calculent à l'aide de l'***algorithme d'Euclide** *et des* **coefficients de Bézout**.
> - *Les calculs de cryptage se feront* **modulo** *n.*
> - *Le déchiffrement fonctionne grâce à une variante du* **petit théorème de Fermat**.

Dans cette section, c'est Bruno qui veut envoyer un message secret à Alice. La processus se décompose ainsi :

1. Alice prépare une clé publique et une clé privée,
2. Bruno utilise la clé publique d'Alice pour crypter son message,
3. Alice reçoit le message crypté et le déchiffre grâce à sa clé privée.

6.1 Calcul de la clé publique et de la clé privée

Choix de deux nombres premiers

Alice effectue, une fois pour toute, les opérations suivantes (en secret) :
- elle choisit deux nombres premiers distincts p et q (dans la pratique ce sont de très grand nombres, jusqu'à des centaines de chiffres),
- Elle calcule $n = p \times q$,
- Elle calcule $\varphi(n) = (p-1) \times (q-1)$.

Exemple 1.
- $p = 5$ et $q = 17$
- $n = p \times q = 85$
- $\varphi(n) = (p-1) \times (q-1) = 64$

Vous noterez que le calcul de $\varphi(n)$ n'est possible que si la décomposition de n sous la forme $p \times q$ est connue. D'où le caractère secret de $\varphi(n)$ même si n est connu de tous.

Exemple 2.
- $p = 101$ et $q = 103$
- $n = p \times q = 10\,403$
- $\varphi(n) = (p-1) \times (q-1) = 10\,200$

Choix d'un exposant et calcul de son inverse

Alice continue :
- elle choisit un exposant e tel que $\text{pgcd}(e, \varphi(n)) = 1$,
- elle calcule l'inverse d de e module $\varphi(n)$: $d \times e \equiv 1 \pmod{\varphi(n)}$. Ce calcul se fait par l'algorithme d'Euclide étendu.

Exemple 1.
- Alice choisit par exemple $e = 5$ et on a bien $\text{pgcd}(e, \varphi(n)) = \text{pgcd}(5, 64) = 1$,
- Alice applique l'algorithme d'Euclide étendu pour calculer les coefficients de Bézout correspondant à $\text{pgcd}(e, \varphi(n)) = 1$. Elle trouve $5 \times 13 + 64 \times (-1) = 1$. Donc $5 \times 13 \equiv 1 \pmod{64}$ et l'inverse de e modulo $\varphi(n)$ est $d = 13$.

Exemple 2.
- Alice choisit par exemple $e = 7$ et on a bien $\text{pgcd}(e, \varphi(n)) = \text{pgcd}(7, 10\,200) = 1$,
- L'algorithme d'Euclide étendu pour $\text{pgcd}(e, \varphi(n)) = 1$ donne $7 \times (-1457) + 10\,200 \times 1 = 1$. Mais $-1457 \equiv 8743 \pmod{\varphi(n)}$, donc pour $d = 8743$ on a $d \times e \equiv 1 \pmod{\varphi(n)}$.

Clé publique

La ***clé publique*** d'Alice est constituée des deux nombres :

n et e

Et comme son nom l'indique Alice communique sa clé publique au monde entier.

Exemple 1. $n = 85$ et $e = 5$

Exemple 2. $n = 10\,403$ et $e = 7$

Clé privée

Alice garde pour elle sa ***clé privée*** :

$$\boxed{d}$$

Alice détruit en secret p, q et $\varphi(n)$ qui ne sont plus utiles. Elle conserve secrètement sa clé privée.

Exemple 1. $d = 13$

Exemple 2. $d = 8743$

6.2 Chiffrement du message

Bruno veut envoyer un message secret à Alice. Il se débrouille pour que son message soit un entier (quitte à découper son texte en bloc et à transformer chaque bloc en un entier).

Message

Le message est un entier m, tel que $0 \leqslant m < n$.

Exemple 1. Bruno veut envoyer le message $m = 10$.

Exemple 2. Bruno veut envoyer le message $m = 1234$.

Message chiffré

Bruno récupère la clé publique d'Alice : n et e avec laquelle il calcule, à l'aide de l'algorithme d'exponentiation rapide, le message chiffré :

$$\boxed{x \equiv m^e \pmod{n}}$$

Il transmet ce message x à Alice

Exemple 1. $m = 10$, $n = 85$ et $e = 5$ donc

$$x \equiv m^e \pmod{n} \equiv 10^5 \pmod{85}$$

On peut ici faire les calculs à la main :

$$\begin{aligned} 10^2 &\equiv 100 \equiv 15 \pmod{85} \\ 10^4 &\equiv (10^2)^2 \equiv 15^2 \equiv 225 \equiv 55 \pmod{85} \\ x &\equiv 10^5 \equiv 10^4 \times 10 \equiv 55 \times 10 \equiv 550 \equiv 40 \pmod{85} \end{aligned}$$

Le message chiffré est donc $x = 40$.

Exemple 2. $m = 1234$, $n = 10\,403$ et $e = 7$ donc

$$x \equiv m^e \pmod{n} \equiv 1234^7 \pmod{10\,403}$$

On utilise l'ordinateur pour obtenir que $x = 10\,378$.

6.3 Déchiffrement du message

Alice reçoit le message x chiffré par Bruno, elle le décrypte à l'aide de sa clé privée d, par l'opération :

$$\boxed{m \equiv x^d \pmod{n}}$$

qui utilise également l'algorithme d'exponentiation rapide.
Nous allons prouver dans le lemme 4, que par cette opération Alice retrouve bien le message original m de Bruno.

Exemple 1. $c = 40$, $d = 13$, $n = 85$ donc

$$x^d \equiv (40)^{13} \pmod{85}.$$

Calculons à la main $40^{13} \equiv \pmod{85}$ on note que $13 = 8+4+1$, donc $40^{13} = 40^8 \times 40^4 \times 40$.

$$\begin{aligned} 40^2 &\equiv 1600 \equiv 70 \pmod{85} \\ 40^4 &\equiv (40^2)^2 \equiv 70^2 \equiv 4900 \equiv 55 \pmod{85} \\ 40^8 &\equiv (40^4)^2 \equiv 55^2 \equiv 3025 \equiv 50 \pmod{85} \end{aligned}$$

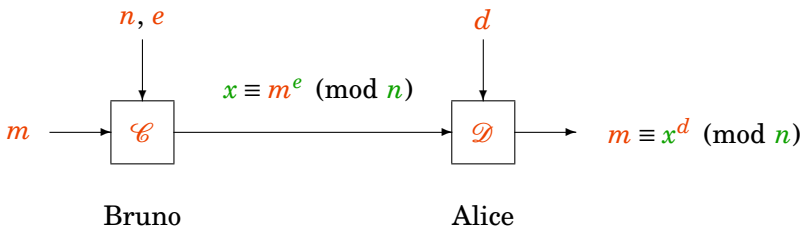
Donc

$$x^d \equiv 40^{13} \equiv 40^8 \times 40^4 \times 40 \equiv 50 \times 55 \times 40 \equiv 10 \pmod{85}$$

qui est bien le message m de Bruno.

Exemple 2. $c = 10\,378$, $d = 8743$, $n = 10\,403$. On calcule par ordinateur $x^d \equiv (10\,378)^{8743} \pmod{10\,403}$ qui vaut exactement le message original de Bruno $m = 1234$.

6.4 Schéma



Clés d'Alice :
- publique : n, e
- privée : d

6.5 Lemme de déchiffrement

Le principe de déchiffrement repose sur le petit théorème de Fermat amélioré.

Lemme 4. Soit d l'inverse de e modulo $\varphi(n)$.

$$\boxed{\text{Si } c \equiv m^e \pmod{n} \text{ alors } m \equiv x^d \pmod{n}.}$$

Ce lemme prouve bien que le message original m de Bruno, chiffré par clé publique d'Alice (e,n) en le message x, peut-être retrouvé par Alice à l'aide de sa clé secrète d.

Démonstration. – Que d soit l'inverse de e modulo $\varphi(n)$ signifie $d \cdot e \equiv 1 \pmod{\varphi(n)}$. Autrement dit, il existe $k \in \mathbb{Z}$ tel que $d \cdot e = 1 + k \cdot \varphi(n)$.

– On rappelle que par le petit théorème de Fermat généralisé : lorsque m et n sont premiers entre eux

$$m^{\varphi(n)} \equiv m^{(p-1)(q-1)} \equiv 1 \pmod{n}$$

– **Premier cas** $\text{pgcd}(m,n) = 1$.
Notons $c \equiv m^e \pmod{n}$ et calculons x^d :

$$x^d \equiv (m^e)^d \equiv m^{e\cdot d} \equiv m^{1+k\cdot\varphi(n)} \equiv m \cdot m^{k\cdot\varphi(n)} \equiv m \cdot (m^{\varphi(n)})^k \equiv m \cdot (1)^k \equiv m \pmod{n}$$

– **Deuxième cas** $\text{pgcd}(m,n) \neq 1$.
Comme n est le produit des deux nombres premiers p et q et que m est strictement plus petit que n alors si m et n ne sont pas premiers entre eux cela implique que p divise m ou bien q divise m (mais pas les deux en même temps). Faisons l'hypothèse $\text{pgcd}(m,n) = p$ et $\text{pgcd}(m,q) = 1$, le cas $\text{pgcd}(m,n) = q$ et $\text{pgcd}(m,p) = 1$ se traiterait de la même manière.
Étudions $(m^e)^d$ à la fois modulo p et modulo q à l'image de ce que nous avons fait dans la preuve du théorème de Fermat amélioré.
  – modulo p : $m \equiv 0 \pmod{p}$ et $(m^e)^d \equiv 0 \pmod{p}$ donc $(m^e)^d \equiv m \pmod{p}$,
  – modulo q : $(m^e)^d \equiv m \times (m^{\varphi(n)})^k \equiv m \times (m^{q-1})^{(p-1)k} \equiv m \pmod{q}$.
  Comme p et q sont deux nombres premiers distincts, ils sont premiers entre eux et on peut écrire comme dans la preuve du petit théorème de Fermat amélioré que

$$(m^e)^d \equiv m \pmod{n}$$

□

6.6 Algorithmes

La mise en œuvre est maintenant très simple. Alice choisit deux nombres premiers p et q et un exposant e.
Voici le calcul de la clé secrète :

rsa.py (1)

```
def cle_privee(p,q,e) :
    n = p * q
    phi = (p-1)*(q-1)
    c,d,dd = euclide_etendu(e,phi)          # Pgcd et coeff de Bézout
    return(d % phi)                         # Bon représentant
```

Le chiffrement d'un message m est possible par tout le monde, connaissant la clé publique (n,e).

rsa.py (2)

```
def codage_rsa(m,n,e):
    return pow(m,e,n)
```

Seule Alice peut déchiffrer le message crypté x, à l'aide de sa clé privée d.

rsa.py (3)

```
def decodage_rsa(x,n,d):
    return pow(x,d,n)
```

Pour continuer...

Bibliographie commentée :

1. **Histoire des codes secrets** de Simon Singh, Le livre de Poche.
   Les codes secrets racontés comme un roman policier. Passionnant. Idéal pour les plus littéraires.
2. **Comprendre les codes secrets** de Pierre Vigoureux, édition Ellipses.
   Un petit livre très clair et très bien écrit, qui présente un panorama complet de la cryptographie sans rentrer dans les détails mathématiques. Idéal pour les esprits logiques.
3. **Codage et cryptographie** de Joan Gómez, édition Le Monde – Images des mathématiques. Un autre petit livre très clair et très bien, un peu de maths, des explications claires et des encarts historiques intéressants.
4. **Introduction à la cryptographie** de Johannes Buchmann, édition Dunod.
   Un livre d'un niveau avancé (troisième année de licence) pour comprendre les méthodes mathématiques de la cryptographie moderne. Idéal pour unifier les points de vue des mathématiques avec l'informatique.
5. **Algèbre - Première année** de Liret et Martinais, édition Dunod.
   Livre qui recouvre tout le programme d'algèbre de la première année, très bien adapté aux étudiants des l'université. Pas de cryptographie.

Auteurs
Arnaud Bodin
François Recher

Algorithmes et mathématiques

Vidéo ■ partie 1. Premiers pas avec Python
Vidéo ■ partie 2. Ecriture des entiers
Vidéo ■ partie 3. Calculs de sinus, cosinus, tangente
Vidéo ■ partie 4. Les réels
Vidéo ■ partie 5. Arithmétique - Algorithmes récursifs
Vidéo ■ partie 6. Polynômes - Complexité d'un algorithme

1 Premiers pas avec Python

Dans cette partie on vérifie d'abord que Python fonctionne, puis on introduira les boucles (`for` et `while`), le test `if ... else ...` et les fonctions.

1.1 Hello world !

Pour commencer testons si tout fonctionne !

Travaux pratiques 1.

1. Définir deux variables prenant les valeurs 3 et 6.
2. Calculer leur somme et leur produit.

Voici à quoi cela ressemble :

hello-world.py

```
>>> a=3
>>> b=6
>>> somme = a+b
>>> print(somme)
9
>>> # Les résultats
>>> print("La somme est", somme)
La somme est 9
>>> produit = a*b
>>> print("Le produit est", produit)
Le produit est 18
```

On retient les choses suivantes :
- On affecte une valeur à une variable par le signe égal =.
- On affiche un message avec la fonction `print()`.
- Lorsque qu'une ligne contient un dièse #, tout ce qui suit est ignoré. Cela permet d'insérer des commentaires, ce qui est essentiel pour relire le code.

Dans la suite on omettra les symboles >>>. Voir plus de détails sur le fonctionnement en fin de section.

1.2 Somme des cubes

Travaux pratiques 2.

1. Pour un entier n fixé, programmer le calcul de la somme $S_n = 1^3 + 2^3 + 3^3 + \cdots + n^3$.
2. Définir une fonction qui pour une valeur n renvoie la somme $\Sigma_n = 1 + 2 + 3 + \cdots + n$.
3. Définir une fonction qui pour une valeur n renvoie S_n.
4. Vérifier, pour les premiers entiers, que $S_n = (\Sigma_n)^2$.

1.

somme-cubes.py (1)

```
n = 10
somme = 0
for i in range(1,n+1):
    somme = somme + i*i*i
print(somme)
```

Voici ce que l'on fait pour calculer S_n avec $n = 10$.
- On affecte d'abord la valeur 0 à la variable `somme`, cela correspond à l'initialisation $S_0 = 0$.
- Nous avons défini une ***boucle*** avec l'instruction *for* qui fait varier i entre 1 et n.
- Nous calculons successivement S_1, S_2,... en utilisant la formule de récurrence $S_i = S_{i-1} + i^3$. Comme nous n'avons pas besoin de conserver toutes les valeurs des S_i alors on garde le même nom pour toutes les sommes, à chaque étape on affecte à `somme` l'ancienne valeur de la somme plus i^3 : `somme = somme + i*i*i`.
- `range(1,n+1)` est l'ensemble des entiers $\{1,2,\ldots,n\}$. C'est bien les entiers **strictement inférieurs à** $n+1$. La raison est que `range(n)` désigne $\{0,1,2,\ldots,n-1\}$ qui contient n éléments.

2. Nous savons que $\Sigma_n = 1+2+3+\cdots+n = \frac{n(n+1)}{2}$ donc nous n'avons pas besoin de faire une boucle :

somme-cubes.py (2)
```
def somme_entiers(n):
    return n*(n+1)/2
```

Une ***fonction*** en informatique est similaire à une fonction mathématique, c'est un objet qui prend en entrée des variables (dites variables formelles ou variables muettes, ici n) et retourne une valeur (un entier, une liste, une chaîne de caractères,... ici $\frac{n(n+1)}{2}$).

3. Voici la fonction qui retourne la somme des cubes :

somme-cubes.py (3)
```
def somme_cubes(n):
    somme = 0
    for i in range(1,n+1):
        somme = somme + i**3
    return somme
```

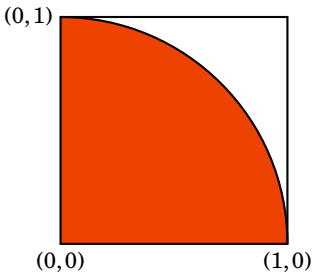
4. Et enfin on vérifie que pour les premiers entiers $S_n = \left(\frac{n(n+1)}{2}\right)^2$, par exemple pour $n = 12$:

somme-cubes.py (4)
```
n = 12
if somme_cubes(n) == (somme_entiers(n)**2):
    print("Pour␣n=", n, "l'assertion␣est␣vraie.")
else:
    print("L'assertion␣est␣fausse␣!")
```

On retient :
- Les puissances se calculent aussi avec `**` : 5^2 s'écrit `5*5` ou `5**2`, 5^3 s'écrit `5*5*5` ou `5**3`,...
- Une fonction se définit par `def ma_fonction(variable):` et se termine par `return resultat`.
- *if condition: ... else: ...* exécute le premier bloc d'instructions si la condition est vraie ; si la condition est fausse cela exécute l'autre bloc.
- Exemple de conditions
  - `a < b` : $a < b$,
  - `a <= b` : $a \leqslant b$,
  - `a == b` : $a = b$,
  - `a != b` : $a \neq b$.
- Attention ! Il est important de comprendre que `a==b` vaut soit vraie ou faux (on compare a et b) alors qu'avec `a=b` on affecte dans a la valeur de b.
- Enfin en `Python` (contrairement aux autres langages) c'est l'indentation (les espaces en début de chaque ligne) qui détermine les blocs d'instructions.

1.3 Calcul de π au hasard

Nous allons voir qu'il est possible de calculer les premières décimales de π par la méthode de Monte-Carlo, c'est à dire avec l'aide du hasard. On considère le carré de coté 1, le cercle de rayon 1 centré à l'origine, d'équation $x^2 + y^2 = 1$, et la portion de disque dans le carré (voir la figure).



Travaux pratiques 3.

1. Calculer l'aire du carré et de la portion de disque.
2. Pour un point (x,y) tiré au hasard dans le carré, quelle est la probabilité que le point soit en fait dans la portion de disque ?
3. Tirer un grand nombre de points au hasard, compter ceux qui sont dans la portion de disque.
4. En déduire les premières décimales de π.

Voici le code :

pi-hasard.py

```
import random             # Module qui génère des nombres aléatoires

Tir = 0                   # Numéro du tir
NbTirDansLeDisque = 0     # Nombre de tirs dans le disque

while (Tir < 1000):
    Tir = Tir + 1
    # On tire au hasard un point (x,y) dans [0,1] x [0,1]
    x = random.random()
    y = random.random()
    if (x*x+y*y <= 1):    # On est dans le disque
        NbTirDansLeDisque = NbTirDansLeDisque + 1

MonPi = 4*NbTirDansLeDisque / Tir
print("Valeur␣expérimentale␣de␣Pi␣:␣%0.3f" %MonPi)
```

Commentaires :

- Un petit calcul prouve que l'aire de la portion de disque est $\frac{\pi}{4}$, l'aire du carré est 1. Donc la probabilité de tomber dans le disque est $\frac{\pi}{4}$.
- Pour tirer un nombre au hasard on utilise une fonction `random()` qui renvoie un nombre réel de l'intervalle $[0,1[$. Bien sûr à chaque appel de la fonction `random()` le nombre obtenu est différent !
- Cette fonction n'est pas connue par défaut de `Python`, il faut lui indiquer le nom du ***module*** où elle se trouve. En début de fichier on ajoute `import random` pour le module qui gère les tirages au hasard. Et pour indiquer qu'une fonction vient d'un module il faut l'appeler par `module.fonction()` donc ici `random.random()` (module et fonction portent ici le même nom !).
- La boucle est *`while condition: ...`* Tant que la condition est vérifiée les instructions de la boucle sont exécutées. Ici `Tir` est le compteur que l'on a initialisé à 0. Ensuite on commence à exécuter la boucle. Bien sûr la première chose que l'on fait dans la boucle est d'incrémenter le compteur `Tir`. On

continue jusqu'à ce que l'on atteigne 999. Pour `Tir`= 1000 la condition n'est plus vraie et le bloc d'instructions du `while` n'est pas exécuté. On passe aux instructions suivantes pour afficher le résultat.
- À chaque tir on teste si on est dans la portion de disque ou pas à l'aide de l'inégalité $x^2+y^2 \leqslant 1$.
- Cette méthode n'est pas très efficace, il faut beaucoup de tirs pour obtenir le deux premières décimales de π.

1.4 Un peu plus sur `Python`

- Le plus surprenant avec `Python` c'est que c'est ***l'indentation*** qui détermine le début et la fin d'un bloc d'instructions. Cela oblige à présenter très soigneusement le code.
- Contrairement à d'autres langages on n'a pas besoin de déclarer le type de variable. Par exemple lorsque l'on initialise une variable par `x=0`, on n'a pas besoin de préciser si x est un entier ou un réel.
- Nous travaillerons avec la version 3 (ou plus) de `Python`, que l'on appelle par `python` ou `python3`. Pour savoir si vous avez la bonne version tester la commande `4/3`. Si la réponse est `1.3333...` alors tout est ok. Par contre avec les versions 1 et 2 de `Python` la réponse est `1` (car il considérait que c'est quotient de la division euclidienne de deux entiers).
- La première façon de lancer `Python` est en ligne de commande, on obtient alors l'invite `>>>` et on tape les commandes.
- Mais le plus pratique est de sauvegarder ses commandes dans un fichier et de faire un appel par `python monfichier.py`
- Vous trouverez sans problème de l'aide et des tutoriels sur internet !

1.5 Mini-exercices

1. Soit le produit $P_n = (1-\frac{1}{2})\times(1-\frac{1}{3})\times(1-\frac{1}{4})\times\cdots\times(1-\frac{1}{n})$. Calculer une valeur approchée de P_n pour les premiers entiers n.
2. Que vaut la somme des entiers i qui apparaissent dans l'instruction `for i in range(1,10)`. Idem pour `for i in range(11)`. Idem pour `for i in range(1,10,2)`. Idem pour `for i in range(0,10,2)`. Idem pour `for i in range(10,0,-1)`.
3. On considère le cube $[0,1]\times[0,1]\times[0,1]$ et la portion de boule de rayon 1 centrée à l'origine incluse dans ce cube. Faire les calculs de probabilité pour un point tiré au hasard dans le cube d'être en fait dans la portion de boule. Faire une fonction pour le vérifier expérimentalement.
4. On lance deux dés. Expérimenter quelle est la probabilité que la somme soit 7, puis 6, puis 3 ? Quelle est la probabilité que l'un des deux dés soit un 6 ? d'avoir un double ? La fonction `randint(a, b)` du module `random` retourne un entier k au hasard, vérifiant $a \leqslant k \leqslant b$.
5. On lance un dé jusqu'à ce que l'on obtienne un 6. En moyenne au bout de combien de lancer s'arrête-t-on ?

2 Écriture des entiers

Nous allons faire un peu d'arithmétique : le quotient de la division euclidienne `//`, le reste `%` (modulo) et nous verrons l'écriture des entiers en base 10 et en base 2. Nous utiliserons aussi la notion de listes et le module `math`.

2.1 Division euclidienne et reste, calcul avec les modulo

La division euclidienne de a par b, avec $a \in \mathbb{Z}$ et $b \in \mathbb{Z}^*$ s'écrit :

$$a = bq + r \quad \text{et} \quad 0 \leqslant r < b$$

où $q \in \mathbb{Z}$ est le ***quotient*** et $r \in \mathbb{N}$ est le ***reste***.
En `Python` le quotient se calcule par : `a // b`. Le reste se calcule par `a % b`. Exemple : `14 // 3` retourne 4 alors que `14 % 3` (lire 14 modulo 3) retourne 2. On a bien $14 = 3\times 4 + 2$.

Les calculs avec les modulos sont très pratiques. Par exemple si l'on souhaite tester si un entier est pair, ou impair cela revient à un test modulo 2. Le code est `if (n%2 == 0): ... else: ...`. Si on besoin de calculer $\cos(n\frac{\pi}{2})$ alors il faut discuter suivant les valeurs de `n%4`.

Appliquons ceci au problème suivant :

Travaux pratiques 4. Combien y-a-t-il d'occurrences du chiffre 1 dans les nombres de 1 à 999 ? Par exemple le chiffre 1 apparaît une fois dans 51 mais deux fois dans 131.

nb-un.py

```
NbDeUn = 0
for N in range(1,999+1):
    ChiffreUnite = N % 10
    ChiffreDizaine = (N // 10) % 10
    ChiffreCentaine = (N // 100) % 10
    if (ChiffreUnite == 1):
        NbDeUn = NbDeUn + 1
    if (ChiffreDizaine == 1):
        NbDeUn = NbDeUn + 1
    if (ChiffreCentaine == 1):
        NbDeUn = NbDeUn + 1
print("Nombre␣d'occurences␣du␣chiffre␣'1'␣:", NbDeUn)
```

Commentaires :
- Comment obtient-on le chiffre des unités d'un entier N ? C'est le reste modulo 10, d'où l'instruction `ChiffreUnite = N % 10`.
- Comment obtient-on le chiffre des dizaines ? C'est plus délicat, on commence par effectuer la division euclidienne de N par 10 (cela revient à supprimer le chiffre des unités, par exemple si $N = 251$ alors `N // 10` retourne 25). Il ne reste plus qu'à calculer le reste modulo 10, (par exemple `(N // 10) % 10` retourne le chiffre des dizaines 5.
- Pour le chiffre des centaines on divise d'abord par 100.

2.2 Écriture des nombres en base 10

L'écriture décimale d'un nombre, c'est associer à un entier N la suite de ses chiffres $[a_0, a_1, \ldots, a_n]$ de sorte que a_i soit le i-ème chiffre de N. C'est-à-dire

$$N = a_n 10^n + a_{n-1} 10^{n-1} + \cdots + a_2 10^2 + a_1 10 + a_0 \quad \text{et } a_i \in \{0, 1, \ldots, 9\}$$

a_0 est le chiffre des unités, a_1 celui des dizaines, a_2 celui des centaines,...

Travaux pratiques 5.

1. Écrire une fonction qui à partir d'une liste $[a_0, a_1, \ldots, a_n]$ calcule l'entier N correspondant.
2. Pour un entier N fixé, combien a-t-il de chiffres ? On pourra s'aider d'une inégalité du type $10^n \leqslant N < 10^{n+1}$.
3. Écrire une fonction qui à partir de N calcule son écriture décimale $[a_0, a_1, \ldots, a_n]$.

Voici le premier algorithme :

decimale.py (1)

```
def chiffres_vers_entier(tab):
    N = 0
    for i in range(len(tab)):
        N = N + tab[i] * (10 ** i)
    return N
```

La formule mathématique est simplement $N = a_n 10^n + a_{n-1}10^{n-1} + \cdots + a_2 10^2 + a_1 10 + a_0$. Par exemple `chiffres_vers_entier([4,3,2,1])` renvoie l'entier 1234.

Expliquons les bases sur les ***listes*** (qui s'appelle aussi des ***tableaux***)
- En `Python` une liste est présentée entre des crochets. Par exemple pour `tab = [4,3,2,1]` alors on accède aux valeurs par `tab[i]` : `tab[0]` vaut 4, `tab[1]` vaut 3, `tab[2]` vaut 2, `tab[3]` vaut 1.
- Pour parcourir les éléments d'un tableau le code est simplement `for x in tab`, x vaut alors successivement 4, 3, 2, 1.
- La longueur du tableau s'obtient par `len(tab)`. Pour notre exemple `len([4,3,2,1])` vaut 4. Pour parcourir toutes les valeurs d'un tableau on peut donc aussi écrire `for i in range(len(tab))`, puis utiliser `tab[i]`, ici i variant ici de 0 à 3.
- La liste vide est seulement notée avec deux crochets : `[]`. Elle est utile pour initialiser une liste.
- Pour ajouter un élément à une liste `tab` existante on utilise la fonction `append`. Par exemple définissons la liste vide `tab=[]`, pour ajouter une valeur à la fin de la liste on saisit : `tab.append(4)`. Maintenant notre liste est [4], elle contient un seul élément. Si on continue avec `tab.append(3)`. Alors maintenant notre liste a deux éléments : [4,3].

Voici l'écriture d'un entier en base 10 :

decimale.py (2)

```
def entier_vers_chiffres(N):
    tab = []
    n = floor(log(N,10))   # le nombre de chiffres est n+1
    for i in range(0,n+1):
        tab.append((N // 10 ** i) % 10)
    return tab
```

Par exemple `entier_vers_chiffres(1234)` renvoie le tableau [4,3,2,1]. Nous avons expliqué tout ce dont nous avions besoin sur les listes au-dessus, expliquons les mathématiques.
- Décomposons $\mathbb{N}^*$ sous la forme $[1,10[\cup [10,100[\cup [100,1000[\cup [1\,000,10\,000[\cup \cdots$ Chaque intervalle est du type $[10^n, 10^{n+1}[$. Pour $N \in \mathbb{N}^*$ il existe donc $n \in \mathbb{N}$ tel que $10^n \leqslant N < 10^{n+1}$. Ce qui indique que le nombre de chiffres de N est $n+1$.
  Par exemple si $N = 1234$ alors $1\,000 = 10^3 \leqslant N < 10^4 = 10\,000$, ainsi $n = 3$ et le nombre de chiffres est 4.
- Comment calculer n à partir de N ? Nous allons utiliser le logarithme décimal $\log_{10}$ qui vérifie $\log_{10}(10) = 1$ et $\log_{10}(10^i) = i$. Le logarithme est une fonction croissante, donc l'inégalité $10^n \leqslant N < 10^{n+1}$ devient $\log_{10}(10^n) \leqslant \log_{10}(N) < \log_{10}(10^{n+1})$. Et donc $n \leqslant \log_{10}(N) < n+1$. Ce qui indique donc que $n = E(\log_{10}(N))$ où $E(x)$ désigne la partie entière d'un réel x.

2.3 Module `math`

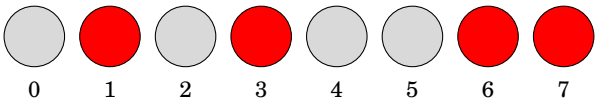
Quelques commentaires informatiques sur un module important pour nous. Les fonctions mathématiques ne sont pas définies par défaut dans `Python` (à part $|x|$ et x^n), il faut faire appel à une librairie spéciale : le module `math` contient les fonctions mathématiques principales.

`abs(x)`	$\lvert x\rvert$
`x ** n`	x^n
`sqrt(x)`	$\sqrt{x}$
`exp(x)`	$\exp x$
`log(x)`	$\ln x$ logarithme népérien
`log(x,10)`	$\log x$ logarithme décimal
`cos(x), sin(x), tan(x)`	$\cos x$, $\sin x$, $\tan x$ en radians
`acos(x), asin(x), atan(x)`	$\arccos x$, $\arcsin x$, $\arctan x$ en radians
`floor(x)`	partie entière $E(x)$:plus grand entier $n \leqslant x$ (*floor* = plancher)
`ceil(x)`	plus petit entier $n \geqslant x$ (*ceil* = plafond)

– Comme on aura souvent besoin de ce module on l'appelle par le code `from math import *`. Cela signifie que l'on importe toutes les fonctions de ce module et qu'en plus on n'a pas besoin de préciser que la fonction vient du module `math`. On peut écrire `cos(3.14)` au lieu `math.cos(3.14)`.
– Dans l'algorithme précédent nous avions utilisé le logarithme décimal `log(x,10)`, ainsi que la partie entière `floor(x)`.

2.4 Écriture des nombres en base 2

On dispose d'une rampe de lumière, chacune des 8 lampes pouvant être allumée (rouge) ou éteinte (gris).



On numérote les lampes de 0 à 7. On souhaite contrôler cette rampe : afficher toutes les combinaisons possibles, faire défiler une combinaison de la gauche à droite (la "chenille"), inverser l'état de toutes les lampes,... Voyons comment l'écriture binaire des nombres peut nous aider. L'***écriture binaire*** d'un nombre c'est son écriture en base 2.
Comment calculer un nombre qui est écrit en binaire ? Le chiffre des "dizaines" correspond à 2 (au lieu de 10), le chiffre des "centaines" à $4 = 2^2$ (au lieu de $100 = 10^2$), le chiffres des "milliers" à $8 = 2^3$ (au lieu de $1000 = 10^3$),... Pour le chiffre des unités cela correspond à $2^0 = 1$ (de même que $10^0 = 1$).
Par exemple 10011_b vaut le nombre 19. Car

$$10011_b = 1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 16 + 2 + 1 = 19.$$

De façon générale tout entier $N \in \mathbb{N}$ s'écrit de manière unique sous la forme

$$N = a_n 2^n + a_{n-1}2^{n-1} + \cdots + a_2 2^2 + a_1 2 + a_0 \quad \text{et} \quad a_i \in \{0,1\}$$

On note alors $N = a_n a_{n-1}\ldots a_1 a_0\ {}_b$ (avec un indice b pour indiquer que c'est son écriture binaire).

Travaux pratiques 6.

1. Écrire une fonction qui à partir d'une liste $[a_0, a_1, \ldots, a_n]$ calcule l'entier N correspondant à l'écriture binaire $a_n a_{n-1}\ldots a_1 a_0\ {}_b$.
2. Écrire une fonction qui à partir de N calcule son écriture binaire sous la forme $[a_0, a_1, \ldots, a_n]$.

La seule différence avec la base 10 c'est que l'on calcule avec des puissances de 2.

binaire.py (1)

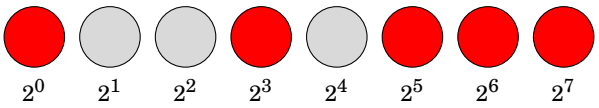
```
def binaire_vers_entier(tab):
    N = 0
    for i in range(len(tab)):
        N = N + tab[i] * (2 ** i)
    return N
```

Idem pour le sens inverse où l'on a besoin du logarithme en base 2, qui vérifie $\log_2(2) = 1$ et $\log_2(2^i) = i$.

binaire.py (2)

```
def entier_vers_binaire(N):
    tab = []
    n = floor(log(N,2))   # le nombre de chiffres est n+1
    for i in range(0,n+1):
        tab.append((N // 2 ** i) % 2)
    return tab
```

Maintenant appliquons ceci à notre problème de lampes. Si une lampe est allumée on lui attribut 1, et si elle est éteinte 0. Pour une rampe de 8 lampes on code $[a_0, a_1, \ldots, a_7]$ l'état des lampes.
Par exemple la configuration suivante :



est codé $[1,0,0,1,0,1,1,1]$ ce qui correspond au nombre binaire $11101001_b = 233$.

Travaux pratiques 7.

1. Faire une boucle qui affiche toutes les combinaisons possibles (pour une taille de rampe donnée).
2. Quelle opération mathématique élémentaire transforme un nombre binaire $a_n \ldots a_1 a_0{}_b$ en $a_n \ldots a_1 a_0 0{}_b$ (décalage vers la gauche et ajout d'un 0 à la fin) ?
3. Soit $N' = a_n a_{n-1} \ldots a_1 a_0 0{}_b$ (une écriture avec $n+2$ chiffres). Quelle est l'écriture binaire de N' (mod 2^{n+1}) ? (C'est une écriture avec $n+1$ chiffres.)
4. En déduire un algorithme qui pour une configuration donnée de la rampe, fait permuter cycliquement (vers la droite) cette configuration. Par exemple $[1,0,1,0,1,1,1,0]$ devient $[0,1,0,1,0,1,1,1]$.
5. Quelle opération mathématique élémentaire permet de passer d'une configuration à son opposée (une lampe éteinte s'allume, et réciproquement). Par exemple si la configuration était $[1,0,1,0,1,1,1,0]$ alors on veut $[0,1,0,1,0,0,0,1]$. (Indication : sur cet exemple calculer les deux nombres correspondants et trouver la relation qui les lie.)

1. Il s'agit d'abord d'afficher les configurations. Par exemple si l'on a 4 lampes alors les configurations sont $[0,0,0,0]$, $[1,0,0,0]$, $[0,1,0,0]$, $[1,1,0,0]$,..., $[1,1,1,1]$. Pour chaque lampe nous avons deux choix (allumé ou éteint), il y a $n+1$ lampes donc un total de 2^{n+1} configurations. Si l'on considère ces configurations comme des nombres écrits en binaire alors l'énumération ci-dessus correspond à compter $0,1,2,3,\ldots,2^{n+1}-1$.
   D'où l'algorithme :

   binaire.py (3)

   ```
   def configurations(n):
       for N in range(2**(n+1)):
           print(entier_vers_binaire_bis(N,n))
   ```

Où `entier_vers_binaire_bis(N,n)` est similaire à `entier_vers_binaire(N)`, mais en affichant aussi les zéros non significatifs, par exemple 7 en binaire s'écrit 111_b, mais codé sur 8 chiffres on ajoute devant des 0 non significatifs : 00000111_b.

2. En écriture décimale, multiplier par 10 revient à décaler le nombre initial et rajouter un zéro. Par exemple $10 \times 19 = 190$. C'est la même chose en binaire ! Multiplier un nombre par 2 revient sur l'écriture à un décalage vers la gauche et ajout d'un zéro sur le chiffre des unités. Exemple : $19 = 10011_b$ et $2 \times 19 = 38$ donc $2 \times 10011_b = 100110_b$.
3. Partant de $N = a_n a_{n-1} \dots a_1 a_0{}_b$. Notons $N' = 2N$, son écriture est $N' = a_n a_{n-1} \dots a_1 a_0 0{}_b$. Alors $N' \pmod{2^{n+1}}$ s'écrit exactement $a_{n-1} a_{n-2} \dots a_1 a_0 0{}_b$ et on ajoute a_n qui est le quotient de N' par 2^{n+1}.

   Preuve : $N' = a_n \cdot 2^{n+1} + a_{n-1} \cdot 2^n + \dots + a_0 \cdot 2$. Donc $N' \pmod{2^{n+1}} = a_{n-1} \cdot 2^n + \dots + a_0 \cdot 2$. Donc $N' \pmod{2^{n+1}} + a_n = a_{n-1} \cdot 2^n + \dots + a_0 \cdot 2 + a_n$.
4. Ainsi l'écriture en binaire de $N' \pmod{2^{n+1}} + a_n$ s'obtient comme permutation circulaire de celle de N. D'où l'algorithme :

binaire.py (4)

```
def decalage(tab):
    N = binaire_vers_entier(tab)
    n = len(tab)-1   # le nombre de chiffres est n+1
    NN = 2*N % 2**(n+1) + 2*N // 2**(n+1)
    return entier_vers_binaire_bis(NN,n)
```

5. On remarque que si l'on a deux configurations opposées alors leur somme vaut $2^{n+1} - 1$: par exemple avec $[1,0,0,1,0,1,1,1]$ et $[0,1,1,0,1,0,0,0]$, les deux nombres associés sont $N = 11101001_b$ et $N' = 00010110_b$ (il s'agit juste de les réécrire de droite à gauche). La somme est $N + N' = 11101001_b + 00010110_b = 11111111_b = 2^8 - 1$. L'addition en écriture binaire se fait de la même façon qu'en écriture décimale et ici il n'y a pas de retenue. Si M est un nombre avec $n+1$ fois le chiffres 1 alors $M + 1 = 2^{n+1}$. Exemple si $M = 11111_b$ alors $M + 1 = 100000_b = 2^5$; ainsi $M = 2^5 - 1$. Donc l'opposé de N est $N' = 2^{n+1} - 1 - N$ (remarquez que dans $\mathbb{Z}/(2^{n+1} - 1)\mathbb{Z}$ alors $N' \equiv -N$).

   Cela conduit à :

binaire.py (5)

```
def inversion(tab):
    N = binaire_vers_entier(tab)
    n = len(tab)-1   # le nombre de chiffres est n+1
    NN = 2**(n+1)-1 - N
    return entier_vers_binaire_bis(NN,n)
```

2.5 Mini-exercices

1. Pour un entier n fixé, combien y-a-t-il d'occurrences du chiffre 1 dans l'écriture des nombres de 1 à n ?
2. Écrire une fonction qui calcule l'écriture décimale d'un entier, sans recourir au log (une boucle `while` est la bienvenue).
3. Écrire un algorithme qui permute cycliquement une configuration de rampe vers la droite.
4. On dispose de $n+1$ lampes, chaque lampe peut s'éclairer de trois couleurs : vert, orange, rouge (dans cet ordre). Trouver toutes les combinaisons possibles. Comment passer toutes les lampes à la couleur suivante ?
5. Générer toutes les matrices 4×4 n'ayant que des 0 et des 1 comme coefficients. On codera une matrice sous la forme de lignes $[[1,1,0,1],[0,0,1,0],[1,1,1,1],[0,1,0,1]]$.

6. On part du point $(0,0) \in \mathbb{Z}^2$. A chaque pas on choisit au hasard un direction Nord, Sud, Est, Ouest. Si on va au Nord alors on ajoute $(0,1)$ à sa position (pour Sud on ajoute $(0,-1)$; pour Est $(1,0)$; pour Ouest $(-1,0)$). Pour un chemin d'une longueur fixée de n pas, coder tous les chemins possibles. Caractériser les chemins qui repassent par l'origine. Calculer la probabilité p_n de repasser par l'origine. Que se passe-t-il lorsque $n \to +\infty$?
7. Écrire une fonction, qui pour un entier N, affiche son écriture en chiffres romains : $M = 1000$, $D = 500$, $C = 100$, $X = 10$, $V = 5$, $I = 1$. Il ne peut y avoir plus de trois symboles identiques à suivre.

3 Calculs de sinus, cosinus, tangente

Le but de cette section est le calcul des sinus, cosinus, et tangente d'un angle par nous même, avec une précision de 8 chiffres après la virgule.

3.1 Calcul de $\operatorname{Arctan} x$

Nous aurons besoin de calculer une fois pour toute $\operatorname{Arctan}(10^{-i})$, pour $i = 0,\dots,8$, c'est-à-dire que l'on cherche les angles $\theta_i \in]-\frac{\pi}{2},\frac{\pi}{2}[$ tels que $\tan\theta_i = 10^{-i}$. Nous allons utiliser la formule :

$$\operatorname{Arctan} x = \sum_{k=0}^{+\infty} (-1)^k \frac{x^{2k+1}}{2k+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

Travaux pratiques 8.

1. Calculer $\operatorname{Arctan} 1$.
2. Calculer $\theta_i = \operatorname{Arctan} 10^{-i}$ (avec 8 chiffres après la virgule) pour $i = 1,\dots,8$.
3. Pour quelles valeurs de i, l'approximation $\operatorname{Arctan} x \simeq x$ était-elle suffisante ?

tangente.py (1)

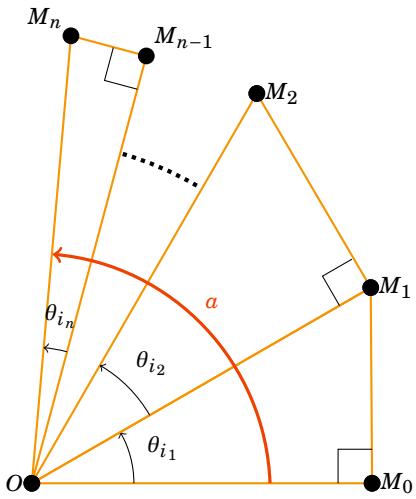
```
def mon_arctan(x,n):
    somme = 0
    for k in range(0,n+1):
        if (k%2 == 0):  # si k est pair signe +
            somme = somme + 1/(2*k+1) * (x ** (2*k+1))
        else:           # si k est impair signe -
            somme = somme - 1/(2*k+1) * (x ** (2*k+1))
    return somme
```

- La série qui permet de calculer $\operatorname{Arctan} x$ est une somme infinie, mais si x est petit alors chacun des termes $(-1)^k \frac{x^{2k+1}}{2k+1}$ est très très petit dès que k devient grand. Par exemple si $0 \leqslant x \leqslant \frac{1}{10}$ alors $x^{2k+1} \leqslant \frac{1}{10^{2k+1}}$ et donc pour $k \geqslant 4$ nous aurons $\left|(-1)^k \frac{x^{2k+1}}{2k+1}\right| < 10^{-9}$. Chacun des termes suivants ne contribue pas aux 8 premiers chiffres après la virgule. Attention : il se pourrait cependant que la somme de beaucoup de termes finissent par y contribuer, mais ce n'est pas le cas ici (c'est un bon exercice de le prouver).
- Dans la pratique on calcule la somme à un certain ordre $2k+1$ jusqu'à ce que les 8 chiffres après la virgules ne bougent plus. Et en fait on s'aperçoit que l'on a seulement besoin d'utiliser $\operatorname{Arctan} x \simeq x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7}$.
- Pour $i \geqslant 4$, $\operatorname{Arctan} x \simeq x$ donne déjà 8 chiffres exacts après la virgule !

On remplit les valeurs des angles θ_i obtenus dans une liste nommée `theta`.

3.2 Calcul de $\tan x$

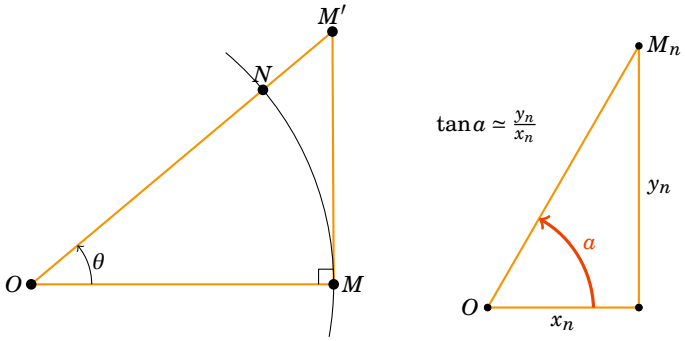
Le principe est le suivant : on connaît un certain nombre d'angles avec leur tangente : les angles θ_i (calculés ci-dessus) avec par définition $\tan\theta_i = 10^{-i}$. Fixons un angle $a \in [0, \frac{\pi}{2}]$. Partant du point $M_0 = (1,0)$, nous allons construire des points $M_1, M_2, \ldots, M_n$ jusqu'à ce que M_n soit (à peu près) sur la demi-droite correspondant à l'angle a. Si M_n a pour coordonnées (x_n, y_n) alors $\tan a = \frac{y_n}{x_n}$. L'angle pour passer d'un point M_k à M_{k+1} est l'un des angles θ_i.



Rappelons que si l'on a un point $M(x,y)$ alors la rotation centrée à l'origine et d'angle θ envoie $M(x,y)$ sur le point $N(x',y')$ avec

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{c'est-à-dire} \quad \begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

Pour un point M, on note M' le point de la demi-droite $[ON)$ tel que les droites (OM) et (MM') soient perpendiculaires en M.



Travaux pratiques 9.

1. (a) Calculer la longueur OM'.
   (b) En déduire les coordonnées de M'.
   (c) Exprimez-les uniquement en fonction de x, y et $\tan\theta$.
2. Faire une boucle qui décompose l'angle a en somme d'angles θ_i (à une précision de 10^{-8} ; avec un minimum d'angles, les angles pouvant se répéter).
3. Partant de $M_0 = (1,0)$ calculer les coordonnées des différents M_k, jusqu'au point $M_n(x_n, y_n)$ correspondant à l'approximation de l'angle a. Renvoyer la valeur $\frac{y_n}{x_n}$ comme approximation de $\tan a$.

Voici les préliminaires mathématiques :
- Dans le triangle rectangle OMM' on a $\cos\theta = \frac{OM}{OM'}$ donc $OM' = \frac{OM}{\cos\theta}$.
- D'autre part comme la rotation d'angle θ conserve les distances alors $OM = ON$. Si les coordonnées de M' sont (x'', y'') alors $x'' = \frac{1}{\cos\theta}x'$ et $y'' = \frac{1}{\cos\theta}y'$.

– Ainsi

$$\begin{cases} x'' = \frac{1}{\cos\theta}x' = \frac{1}{\cos\theta}\big(x\cos\theta - y\sin\theta\big) = x - y\tan\theta \\ y'' = \frac{1}{\cos\theta}y' = \frac{1}{\cos\theta}\big(x\sin\theta + y\cos\theta\big) = x\tan\theta + y \end{cases}$$

Autrement dit :

$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Voici une boucle simple pour décomposer l'angle θ : on commence par retirer le plus grand angle θ_0 autant de fois que l'on peut, lorsque ce n'est plus possible on passe à l'angle θ_1,...

tangente.py (2)

```
i = 0
while (a > precision):      # boucle tant que la precision pas atteinte
    while (a < theta[i]):   # choix du bon angle theta_i à soustraire
        i = i+1
    a = a - theta[i]        # on retire l'angle theta_i et on recommence
```

Ici `precision` est la précision souhaité (pour nous 10^{-9}). Et le tableau `theta` contient les valeurs des angles θ_i.

Posons $x_0 = 1$, $y_0 = 0$ et $M_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$. Alors on définit par récurrence $M_{k+1} = P(\theta_i)\cdot M_k$ où $P(\theta) = \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix}$.
Les θ_i sont ceux apparaissant dans la décomposition de l'angle en somme de θ_i, donc on connaît $\tan\theta_i = 10^{-i}$. Ainsi si l'on passe d'un point M_k à M_{k+1} par un angle θ_i on a simplement :

$$\begin{cases} x_{k+1} = x_k - y_k \cdot 10^{-i} \\ y_{k+1} = x_k \cdot 10^{-i} + y_k \end{cases}$$

La valeur $\frac{y_n}{x_n}$ est la tangente de la somme des angles θ_i, donc une approximation de $\tan\alpha$.
Le code est maintenant le suivant.

tangente.py (3)

```
def ma_tan(a):
    precision = 10**(-9)
    i = 0 ; x = 1 ; y = 0
    while (a > precision):
        while (a < theta[i]):
            i = i+1
        newa = a - theta[i]       # on retire l'angle theta_i
        newx = x - (10**(-i))*y   # on calcule le nouveau point
        newy = (10**(-i))*x + y
        x = newx
        y = newy
        a = newa
    return y/x                    # on renvoie la tangente
```

Commentaires pour conclure :

– En théorie il ne faut pas confondre «précision» et «nombre de chiffres exacts après la virgule». Par exemple 0.999 est une valeur approchée de 1 à 10^{-3} près, mais aucun chiffre après la virgule n'est exact. Dans la pratique c'est la précision qui importe plus que le nombre de chiffres exacts.

– Notez à quel point les opérations du calcul de $\tan x$ sont simples : il n'y a quasiment que des additions à effectuer. Par exemple l'opération $x_{k+1} = x_k - y_k \cdot 10^{-i}$ peut être fait à la main : multiplier par 10^{-i} c'est juste décaler la virgule à droite de i chiffres, puis on additionne. C'est cet algorithme «CORDIC» qui est implémenté dans les calculatrices, car il nécessite très peu de ressources. Bien sûr, si les nombres sont codés en binaire on remplace les 10^{-i} par 2^{-i} pour n'avoir qu'à faire des décalages à droite.

3.3 Calcul de $\sin x$ et $\cos x$

Travaux pratiques 10. Pour $0 \leqslant x \leqslant \frac{\pi}{2}$, calculer $\sin x$ et $\cos x$ en fonction de $\tan x$. En déduire comment calculer les sinus et cosinus de x.

Solution : On sait $\cos^2 + \sin^2 x = 1$, donc en divisant par $\cos^2 x$ on trouve $1 + \tan^2 x = \frac{1}{\cos^2 x}$. On en déduit que pour $0 \leqslant x \leqslant \frac{\pi}{2}$ $\cos x = \frac{1}{\sqrt{1+\tan^2 x}}$. On trouve de même $\sin x = \frac{\tan x}{\sqrt{1+\tan^2 x}}$.
Donc une fois que l'on a calculé $\tan x$ on en déduit $\sin x$ et $\cos x$ par un calcul de racine carrée. Attention c'est valide car x est compris entre 0 et $\frac{\pi}{2}$. Pour un x quelconque il faut se ramener par les formules trigonométriques à l'intervalle $[0, \frac{\pi}{2}]$.

3.4 Mini-exercices

1. On dispose de billets de 1, 5, 20 et 100 euros. Trouvez la façon de payer une somme de n euros avec le minimum de billets.
2. Faire un programme qui pour **n'importe quel** $x \in \mathbb{R}$, calcule $\sin x$, $\cos x$, $\tan x$.
3. Pour $t = \tan \frac{x}{2}$ montrer que $\tan x = \frac{2t}{1-t^2}$. En déduire une fonction qui calcule $\tan x$. (Utiliser que pour x assez petit $\tan x \simeq x$).
4. Modifier l'algorithme de la tangente pour qu'il calcule aussi directement le sinus et le cosinus.

4 Les réels

Dans cette partie nous allons voir différentes façons de calculer la constante γ d'Euler. C'est un nombre assez mystérieux car personne ne sait si γ est un nombre rationnel ou irrationnel. Notre objectif est d'avoir le plus de décimales possibles après la virgule en un minimum d'étapes. Nous verrons ensuite comment les ordinateurs stockent les réels et les problèmes que cela engendre.

4.1 Constante γ d'Euler

Considérons la ***suite harmonique*** :

$$H_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

et définissons

$$u_n = H_n - \ln n.$$

Cette suite (u_n) admet une limite lorsque $n \to +\infty$: c'est la ***constante γ d'Euler***.

Travaux pratiques 11.

1. Calculer les premières décimales de γ. Sachant que $u_n - \gamma \sim \frac{1}{2n}$, combien de décimales exactes peut-on espérer avoir obtenues ?
2. On considère $v_n = H_n - \ln\left(n + \frac{1}{2} + \frac{1}{24n}\right)$. Sachant $v_n - \gamma \sim -\frac{1}{48n^3}$, calculer davantage de décimales.

euler.py (1)

```
def euler1(n):
    somme = 0
    for i in range(n,0,-1):
        somme = somme + 1/i
    return somme - log(n)
```

euler.py (2)

```
def euler2(n):
    somme = 0
    for i in range(n,0,-1):
        somme = somme + 1/i
    return somme - log(n+1/2+1/(24*n))
```

Vous remarquez que la somme est calculée à partir de la fin. Nous expliquerons pourquoi en fin de section.

4.2 1000 décimales de la constante d'Euler

Il y a deux techniques pour obtenir plus de décimales : (i) pousser plus loin les itérations, mais pour avoir 1000 décimales de γ les méthodes précédentes sont insuffisantes ; (ii) trouver une méthode encore plus efficace. C'est ce que nous allons voir avec la méthode de Bessel modifiée.
Soit

$$w_n = \frac{A_n}{B_n} - \ln n \quad \text{avec} \quad A_n = \sum_{k=1}^{E(\alpha n)} \left(\frac{n^k}{k!}\right)^2 H_k \quad \text{et} \quad B_n = \sum_{k=0}^{E(\alpha n)} \left(\frac{n^k}{k!}\right)^2$$

où $\alpha = 3.59112147\ldots$ est la solution de $\alpha(\ln\alpha - 1) = 1$ et $E(x)$ désigne la partie entière. Alors

$$|w_n - \gamma| \leqslant \frac{C}{e^{4n}}$$

où C est une constante (non connue).

Travaux pratiques 12.

1. Programmer cette méthode.
2. Combien d'itérations faut-il pour obtenir 1000 décimales ?
3. Utiliser le module `decimal` pour les calculer.

Voici le code :

euler.py (3)

```
def euler3(n):
    alpha = 3.59112147
    N = floor(alpha*n)      # Borne des sommes
    A = 0 ; B = 0
    H = 0
    for k in range(1,N+1):
        c = ( (n**k)/factorial(k) ) ** 2   # Coefficient commun
        H = H + 1/k                        # Somme harmonique
        A = A + c*H
        B = B + c
    return A/B - log(n)
```

Pour obtenir N décimales il faut résoudre l'inéquation $\frac{C}{e^{4n}} \leqslant \frac{1}{10^N}$. On «passe au log» pour obtenir $n \geqslant \frac{N\ln(10)+\ln(C)}{4}$. On ne connaît pas C mais ce n'est pas si important. Moralement pour une itération de plus on obtient (à peu près) une décimale de plus (c'est-à-dire un facteur 10 sur la précision !). Pour $n \geqslant 800$ on obtient 1000 décimales exactes de la constante d'Euler :

0,
5772156649015328606065120900824024310421593359399235988057672348848677267776646709369470632917467495
1463144724980708248096050401448654283622417399764492353625350033374293733773767394279259525824709491
6008735203948165670853233151776611528621199501507984793745085705740029921354786146694029604325421519
0587755352673313992540129674205137541395491116851028079842348775872050384310939973613725530608893312
6760017247953783675927135157722610273492913940798430103417771778088154957066107501016191663340152278
9358679654972520362128792265559536696281763887927268013243101047650596370394739495763890657296792960
1009015125195950922243501409349871228247949747195646976318506676129063811051824197444867836380861749
4551698927923018773910729457815543160050021828440960537724342032854783670151773943987003023703395183
2869000155819398804270741154222781971652301107356583339673487176504919418123000406546931429992977795
6
9303100503086303418569803231083691640025892970890985486825777364288253954925873629596133298574739302

Pour obtenir plus de décimales que la précision standard de `Python`, il faut utiliser le module `decimal` qui permet de travailler avec une précision arbitraire fixée.

4.3 Un peu de réalité

En mathématique un réel est un élément de $\mathbb{R}$ et son écriture décimale est souvent infinie après la virgule : par exemple $\pi = 3,14159265\ldots$ Mais bien sûr un ordinateur ne peut pas coder une infinité d'informations. Ce qui se rapproche d'un réel est un ***nombre flottant*** dont l'écriture est :

$$\underbrace{\pm 1,234567890123456789}_{\text{mantisse}}\ e\ \underbrace{\pm 123}_{\text{exposant}}$$

pour $\pm 1,234\ldots \times 10^{\pm 123}$. La ***mantisse*** est un nombre décimal (positif ou négatif) appartenant à $[1,10[$ et l'exposant est un entier (lui aussi positif ou négatif). En `Python` la mantisse à une précision de 16 chiffres après la virgule.
Cette réalité informatique fait que des erreurs de calculs peuvent apparaître même avec des opérations simples. Pour voir un exemple de problème faites ceci :

Travaux pratiques 13. Poser $x = 10^{-16}$, $y = x + 1$, $z = y - 1$. Que vaut z pour `Python` ?

Comme `Python` est très précis nous allons faire une routine qui permet de limiter drastiquement le nombre de chiffres et mettre en évidence les erreurs de calculs.

Travaux pratiques 14.

1. Calculer l'exposant d'un nombre réel. Calculer la mantisse.
2. Faire une fonction qui ne conserve que 6 chiffres d'un nombre (6 chiffres en tout : avant + après la virgule, exemple $123,456789$ devient $123,456$).

Voici le code :

reels.py (1)

```
precision = 6                  # Nombre de décimales conservées
def tronquer(x):
    n = floor(log(x,10))                        # Exposant
    m = floor( x * 10 ** (precision-1 - n))     # Mantisse
    return m * 10 ** (-precision+1+n)           # Nombre tronqué
```

Comme on l'a déjà vu auparavant l'exposant se récupère à l'aide du logarithme en base 10. Et pour tronquer un nombre avec 6 chiffres, on commence par le décaler vers la gauche pour obtenir 6 chiffres avant la virgule ($123,456789$ devient $123456,789$) il ne reste plus qu'à prendre la partie entière (123456) et le redécaler vers la droite (pour obtenir $123,456$).

Absorption

Travaux pratiques 15.

1. Calculer `tronquer(1234.56 + 0.007)`.
2. Expliquer.

Chacun des nombres 1234,56 et 0,007 est bien un nombre s'écrivant avec moins de 6 décimales mais leur somme 1234,567 a besoin d'une décimale de plus, l'ordinateur ne retient pas la 7-ème décimale et ainsi le résultat obtenu est 1234,56. Le 0,007 n'apparaît pas dans le résultat : il a été victime d'une ***absorption***.

Élimination

Travaux pratiques 16.

1. Soient $x = 1234,8777$, $y = 1212,2222$. Calculer $x - y$ à la main. Comment se calcule la différence $x - y$ avec notre précision de 6 chiffres ?
2. Expliquer la différence.

Comme $x - y = 22,6555$ qui n'a que 6 chiffres alors on peut penser que l'ordinateur va obtenir ce résultat. Il n'en est rien, l'ordinateur ne stocke pas x mais `tronquer(x)`, idem pour y. Donc l'ordinateur effectue en fait le calcul suivant : `tronquer(tronquer(x)-tronquer(y))`, il calcule donc $1234,87 - 1212,22 = 22,65$. Quel est le problème ? C'est qu'ensuite l'utilisateur considère –à tort– que le résultat est calculé avec une précision de 6 chiffres. Donc on peut penser que le résultat est 22,6500 mais les 2 derniers chiffres sont une pure invention.
C'est un phénomène d'***élimination***. Lorsque l'on calcule la différence de deux nombres proches, le résultat a en fait une précision moindre. Cela peut être encore plus dramatique avec l'exemple $\delta = 1234,569 - 1234,55$ la différence est 0,01900 alors que l'ordinateur retournera 0,01000. Il y a presque un facteur deux, et on aura des problèmes si l'on a besoin de diviser par δ.

Signalons au passage une erreur d'interprétation fréquente : ne pas confondre la ***précision*** d'affichage (exemple : on calcule avec 10 chiffres après la virgule) avec l'***exactitude*** du résultat (combien de décimales sont vraiment exactes ?).

Conversion binaire – décimale

Enfin le problème le plus troublant est que les nombres flottants sont stockés en écriture binaire et pas en écriture décimale.

Travaux pratiques 17. Effectuer les commandes suivantes et constater !

1. `sum = 0` puis `for i in range(10): sum = sum + 0.1`. Que vaut `sum` ?
2. `0.1 + 0.1 == 0.2` et `0.1 + 0.1 + 0.1 == 0.3`
3. `x = 0.2 ; print("0.2␣en␣Python␣=␣%.25f" %x)`

La raison est simple mais néanmoins troublante. L'ordinateur ne stocke pas 0,1, ni 0,2 en mémoire mais le nombre en écriture binaire qui s'en rapproche le plus.
En écriture décimale, il est impossible de coder $1/3 = 0,3333\ldots$ avec un nombre fini de chiffres après la virgule. Il en va de même ici : l'ordinateur ne peut pas stocker exactement 0,2. Il stocke un nombre en écriture binaire qui s'en rapproche le plus ; lorsqu'on lui demande d'afficher le nombre stocké, il retourne l'écriture décimale qui se rapproche le plus du nombre stocké, mais ce n'est plus 0,2, mais un nombre très très proche :

`0.20000000000000000111022302...`

4.4 Somme des inverses des carrés

Voyons une situation concrète où ces problèmes apparaissent.

Travaux pratiques 18.

1. Faire une fonction qui calcule la somme $S_n = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$.
2. Faire une fonction qui calcule cette somme mais en utilisant seulement une écriture décimale à 6 chiffres (à l'aide de la fonction `tronquer()` vue au-dessus).
3. Reprendre cette dernière fonction, mais en commençant la somme par les plus petits termes.
4. Comparez le deux dernières méthodes, justifier et conclure.

La première fonction ne pose aucun problème et utilise toute la précision de `Python`.
Dans la seconde on doit, à chaque calcul, limiter notre précision à 6 chiffres (ici 1 avant la virgule et 5 après).

reels.py (2)

```
def somme_inverse_carres_tronq(n):
  somme = 0
  for i in range(1,n+1):
      somme = tronquer(somme + tronquer(1/(i*i)))
  return somme
```

Il est préférable de commencer la somme par la fin :

reels.py (3)

```
def somme_inverse_carres_tronq_inv(n):
  somme = 0
  for i in range(n,0,-1):
      somme = tronquer(somme + tronquer(1/(i*i)))
  return somme
```

Par exemple pour $n = 100\,000$ l'algorithme `somme_inverse_carres_tronq()` (avec écriture tronquée, sommé dans l'ordre) retourne 1,64038 alors que l'algorithme `somme_inverse_carres_tronq_inv()` (avec la somme dans l'ordre inverse) on obtient 1,64490. Avec une précision maximale et n très grand on doit obtenir 1,64493… (en fait c'est $\frac{\pi^2}{6}$).
Notez que faire grandir n pour l'algorithme `somme_inverse_carres_tronq()` n'y changera rien, il bloque à 2 décimales exactes après la virgule : 1,64038 ! La raison est un phénomène d'absorption : on rajoute des termes très petits devant une somme qui vaut plus de 1. Alors que si l'on part des termes petits, on ajoute des termes petits à une somme petite, on garde donc un maximum de décimales valides avant de terminer par les plus hautes valeurs.

4.5 Mini-exercices

1. Écrire une fonction qui approxime la constante α qui vérifie $\alpha(\ln\alpha - 1) = 1$. Pour cela poser $f(x) = x(\ln x - 1) - 1$ et appliquer la méthode de Newton : fixer u_0 (par exemple ici $u_0 = 4$) et $u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$.
2. Pour chacune des trois méthodes, calculer le nombre approximatif d'itérations nécessaires pour obtenir 100 décimales de la constante γ d'Euler.
3. Notons $C_n = \frac{1}{4n}\sum_{k=0}^{2n} \frac{[(2k)!]^3}{(k!)^4(16n)^{2k}}$. La formule de Brent-McMillan affirme $\gamma = \frac{A_n}{B_n} - \frac{C_n}{B_n^2} - \ln n + O(\frac{1}{e^{8n}})$ où cette fois les sommations pour A_n et B_n vont jusqu'à $E(\beta n)$ avec $\beta = 4{,}970625759\ldots$ la solution de $\beta(\ln\beta - 1) = 3$. La notation $O(\frac{1}{e^{8n}})$ indique que l'erreur est $\leqslant \frac{C}{e^{8n}}$ pour une certaine constante C. Mettre en œuvre cette formule. En 1999 cette formule a permis de calculer 100 millions de décimales. Combien a-t-il fallu d'itérations ?

4. Faire une fonction qui renvoie le terme u_n de la suite définie par $u_0 = \frac{1}{3}$ et $u_{n+1} = 4u_n - 1$. Que vaut u_{100} ? Faire l'étude mathématique et commenter.

5 Arithmétique – Algorithmes récursifs

Nous allons présenter quelques algorithmes élémentaires en lien avec l'arithmétique. Nous en profitons pour présenter une façon complètement différente d'écrire des algorithmes : les fonctions récursives.

5.1 Algorithmes récursifs

Voici un algorithme très classique :

recursif.py (1)

```
def factorielle_classique(n):
    produit = 1
    for i in range(1,n+1):
        produit = i * produit
    return produit
```

Voyons comment fonctionne cette boucle. On initialise la variable `produit` à 1, on fait varier un indice i de 1 à n. À chaque étape on multiplie `produit` par i et on affecte le résultat dans `produit`. Par exemple si $n = 5$ alors la variable `produit` s'initialise à 1, puis lorsque i varie la variable produit devient $1 \times 1 = 1$, $2 \times 1 = 2$, $3 \times 2 = 6$, $4 \times 6 = 24$, $5 \times 24 = 120$. Vous avez bien sûr reconnus le calcul de 5!

Étudions un autre algorithme.

recursif.py (2)

```
def factorielle(n):
    if (n==1):
        return 1
    else:
        return n * factorielle(n-1)
```

Que fait cet algorithme ? Voyons cela pour $n = 5$. Pour $n = 5$ la condition du «si» (if) n'est pas vérifiée donc on passe directement au «sinon» (else). Donc `factorielle(5)` renvoie comme résultat : `5 * factorielle(4)`. On a plus ou moins progressé : le calcul n'est pas fini car on ne connaît pas encore `factorielle(4)` mais on s'est ramené à un calcul au rang précédent, et on itère :
`factorielle(5) = 5 * factorielle(4) = 5 * 4 * factorielle(3) = 5 * 4 * 3 * factorielle(2)`
et enfin `factorielle(5) = 5 * 4 * 3 * 2 * factorielle(1)`. Pour `factorielle(1)` la condition du `if (n==1)` est vérifiée et alors `factorielle(1)=1`. Le bilan est donc que `factorielle(5) = 5 * 4 * 3 * 2 * 1` c'est bien 5!
Une fonction qui lorsque elle s'exécute s'appelle elle-même est une ***fonction récursive***. Il y a une analogie très forte avec la récurrence. Par exemple on peut définir la suite des factorielles ainsi :

$$u_1 = 1 \quad \text{et} \quad u_n = n \times u_{n-1} \text{ si } n \geqslant 2.$$

Nous avons ici $u_n = n!$ pour tout $n \geqslant 1$.
Comme pour la récurrence une fonction récursive comporte une étape d'***initialisation*** (ici `if (n==1): return 1` correspondant à $u_1 = 1$) et une étape d'***hérédité*** (ici `return n * factorielle(n-1)` correspondant à $u_n = n \times u_{n-1}$).

On peut même faire deux appels à la fonction :

recursif.py (3)

```
def fibonacci(n):
    if (n==0) or (n==1):
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
```

Faites-le calcul de `fibonacci(5)`. Voici la version mathématique des nombres de Fibonacci.

$$F_0 = 1,\ F_1 = 1 \quad \text{et} \quad F_n = F_{n-1} + F_{n-2} \quad \text{si } n \geqslant 2.$$

On obtient un nombre en additionnant les deux nombres des rangs précédents :

$$1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21 \quad 34 \quad \ldots$$

5.2 L'algorithme d'Euclide

L'algorithme d'Euclide est basé sur le principe suivant

$$\text{si } b|a \text{ alors } \operatorname{pgcd}(a,b) = b \qquad \text{sinon } \operatorname{pgcd}(a,b) = \operatorname{pgcd}(b, a \bmod b)$$

Travaux pratiques 19.

1. Créer une fonction récursive `pgcd(a,b)` qui calcule le pgcd.
2. On note p_n la probabilité que deux entiers a,b tirés au hasard dans $1,2,\ldots,n$ soient premiers entre eux. Faire une fonction qui approxime p_n. Lorsque n devient grand, comparer p_n et $\frac{6}{\pi^2}$.

Voici le code pour l'algorithme d'Euclide récursif. Notez à quel point le code est succinct et épuré !

arith.py (1)

```
def pgcd(a,b):
    if a%b == 0:
        return b
    else:
        return pgcd(b, a%b)
```

Deux entiers a,b sont premiers entre eux ssi $\operatorname{pgcd}(a,b) = 1$, donc voici l'algorithme :

arith.py (2)

```
def nb_premiers_entre_eux(n,nbtirages):
    i = 1
    nbpremiers = 0
    while i <= nbtirages:
        i = i+1
        a = random.randint(1,n)
        b = random.randint(1,n)
        if pgcd(a,b)==1:
            nbpremiers = nbpremiers + 1
    return nbpremiers
```

On tire au hasard deux entiers a et b entre 1 et n et on effectue cette opération `nbtirages` fois. Par exemple entre 1 et 1000 si l'on effectue 10000 tirage on trouve une probabilité mesurée par `nbpremiers/nbtirages` de $0,60\ldots$ (les décimales d'après dépendent des tirages).
Lorsque n tend vers $+\infty$ alors $p_n \to \frac{6}{\pi^2} = 0.607927\ldots$ et on dit souvent que : «la probabilité que deux entiers tirés au hasard soient premiers entre eux est $\frac{6}{\pi^2}$.»

Commentaires sur les algorithmes récursifs :
- Les algorithmes récursifs ont souvent un code très court, et proche de la formulation mathématique lorsque l'on a une relation de récurrence.
- Selon le langage ou la fonction programmée il peut y avoir des problèmes de mémoire (si par exemple pour calculer 5! l'ordinateur a besoin de stocker 4! pour lequel il a besoin de stocker 3!...).
- Il est important de bien réfléchir à la condition initiale (qui est en fait celle qui termine l'algorithme) et à la récurrence sous peine d'avoir une fonction qui boucle indéfiniment !
- Il n'existe pas des algorithmes récursifs pour tout (voir par exemple les nombres premiers) mais ils apparaissent beaucoup dans les algorithmes de tris. Autre exemple : la dichotomie se programme très bien par une fonction récursive.

5.3 Nombres premiers

Les nombres premiers offrent peu de place aux algorithmes récursifs car il n'y a pas de lien de récurrence entre les nombres premiers.

Travaux pratiques 20.

1. Écrire une fonction qui détecte si un nombre n est premier ou pas en testant s'il existe des entiers k qui divise n. (On se limitera aux entiers $2 \leqslant k \leqslant \sqrt{n}$, pourquoi ?).
2. Faire un algorithme pour le crible d'Eratosthène : écrire tous les entiers de 2 à n, conserver 2 (qui est premier) et barrer tous les multiples suivants de 2. Le premier nombre non barré (c'est 3) est premier. Barrer tous les multiples suivants de 3,...
3. Dessiner la spirale d'Ulam : on place les nombres entiers en spirale, et on colorie en rouge les nombres premiers.

$$\begin{array}{cccc} & & \ldots & \ldots \\ & & & \vdots \\ 5 & 4 & 3 & \\ 6 & 1 & 2 & 11 \\ 7 & 8 & 9 & 10 \end{array}$$

1. Si n n'est pas premier alors $n = a \times b$ avec $a, b \geqslant 2$. Il est clair que soit $a \leqslant \sqrt{n}$ ou bien $b \leqslant \sqrt{n}$ (sinon $n = a \times b > n$). Donc il suffit de tester les diviseurs $2 \leqslant k \leqslant \sqrt{n}$. D'où l'algorithme :

arith.py (3)

```
def est_premier(n):
    if (n<=1): return False
    k = 2
    while k*k <= n:
        if n%k==0:
            return False
        else:
            k = k +1
    return True
```

Notez qu'il vaut mieux écrire la condition `k*k <= n` plutôt que `k <= sqrt(n)` : il est beaucoup plus rapide de calculer le carré d'un entier plutôt qu'extraire une racine carrée.
Nous avons utilisé un nouveau type de variable : un ***booléen*** est une variable qui ne peut prendre que deux états Vrai ou Faux (ici *True* or *False*, souvent codé 1 et 0). Ainsi `est_premier(13)` renvoie `True`, alors que `est_premier(14)` renvoie `False`.

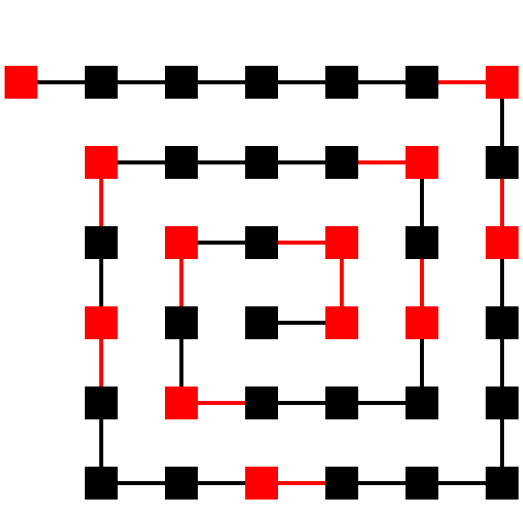
2. Pour le crible d'Eratosthène le plus dur est de trouver le bon codage de l'information.

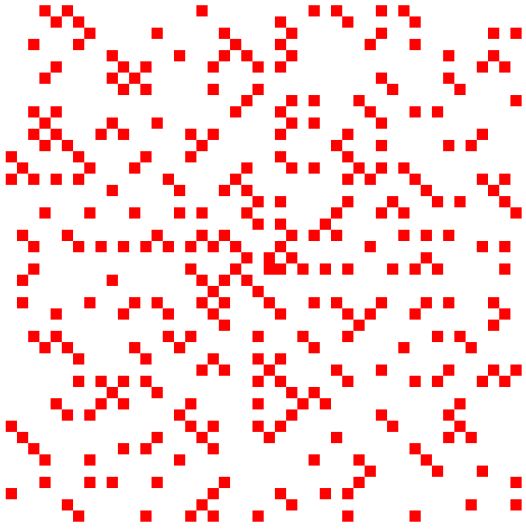
arith.py (4)

```
def eratosthene(n):
    liste_entiers = list(range(n+1))  # tous les entiers
    liste_entiers[1] = 0              # 1 n'est pas premier
    k = 2                             # on commence par les multiples de 2
    while k*k <= n:
        if liste_entiers[k] != 0: # si le nombre k n'est pas barré
            i = k                 # les i sont les multiples de k
            while i <= n-k:
                i = i+k
                liste_entiers[i] = 0  # multiples de k : pas premiers
        k = k +1
    liste_premiers = [k for k in liste_entiers if k !=0]  # efface les 0
    return liste_premiers
```

Ici on commence par faire un tableau contenant les entiers `[0,1,2,3,4,5,6,7,8,9,10,11,12,13,...]`. Pour signifier qu'un nombre n'est pas premier ou remplace l'entier par `0`. Comme 1 n'est pas un nombre premier : on le remplace par 0. Puis on fait une boucle, on part de 2 et on remplace tous les autres multiples de 2 par 0 : la liste est maintenant : `[0,0,2,3,0,5,0,7,0,9,0,11,0,13,...]`. Le premiers nombre après 2 est 3 c'est donc un nombre premier. (car s'il n'a aucun diviseur autre que 1 et lui-même car sinon il aurait été rayé). On garde 3 et remplace tous les autres multiples de 3 par 0. La liste est maintenant : `[0,0,2,3,0,5,0,7,0,0,0,11,0,13,...]`. On itère ainsi, à la fin on efface les zéros pour obtenir : `[2,3,5,7,11,13,...]`.

3. Pour la spirale d'Ulam la seule difficulté est de placer les entiers sur une spirale, voici le résultat.





À gauche le début de la spirale (de $n = 1$ à 37) en rouge les nombres premiers (en noir les nombres non premiers) ; à droite le motif obtenu jusqu'à de grandes valeurs (en blanc les nombres non premiers).

5.4 Mini-exercices

1. Écrire une version itérative et une version récursive pour les fonctions suivantes : (a) la somme des carrés des entiers de 1 à n ; (b) 2^n (sans utiliser d'exposant) ; (c) la partie entière d'un réel $x \geqslant 0$; (d) le quotient de la division euclidienne de a par b (avec $a \in \mathbb{N}$, $b \in \mathbb{N}^*$) ; (e) le reste de cette division euclidienne (sans utiliser les commandes % ni //).

2. Écrire une version itérative de la suite de Fibonacci.
3. Écrire une version itérative de l'algorithme d'Euclide. Faire une version qui calcule les coefficients de Bézout.
4. Écrire une fonction itérative, puis récursive, qui pour un entier n renvoie la liste de ses diviseurs. Dessiner une spirale d'Ulam, dont l'intensité de la couleur dépend du nombre de diviseurs.
5. Une suite de Syracuse est définie ainsi : partant d'un entier s'il est pair on le divise par deux, s'il est impair on le multiplie par 3 et on ajoute 1. On itère ce processus. Quelle conjecture peut-on faire sur cette suite ?
6. Dessiner le triangle de Pascal $\begin{matrix} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ & \cdots & & & & \cdots & \end{matrix}$ Ensuite effacer tous les coefficients pairs (ou mieux : remplacer les coefficients pairs par un carré blanc et les coefficients impairs par un carré rouge). Quelle figure reconnaissez-vous ?

6 Polynômes – Complexité d'un algorithme

Nous allons étudier la complexité des algorithmes à travers l'exemple des polynômes.

6.1 Qu'est-ce qu'un algorithme ?

Qu'est ce qu'un algorithme ? Un algorithme est une succession d'instructions qui renvoie un résultat. Pour être vraiment un algorithme on doit justifier que le résultat retourné est ***exact*** (le programme fait bien ce que l'on souhaite) et ceci en un ***nombre fini d'étapes*** (cela renvoie le résultat en temps fini).
Maintenant certains algorithmes peuvent être plus rapides que d'autres. C'est souvent le temps de calcul qui est le principal critère, mais cela dépend du langage et de la machine utilisée. Il existe une manière plus mathématique de faire : la ***complexité*** d'un algorithme c'est le nombre d'opérations élémentaires à effectuer.
Ces opérations peuvent être le nombre d'opérations au niveau du processeur, mais pour nous ce sera le nombre d'additions $+$, le nombre de multiplications $\times$ à effectuer. Pour certains algorithmes la vitesse d'exécution n'est pas le seul paramètre mais aussi la taille de la mémoire occupée.

6.2 Polynômes

Travaux pratiques 21. On code un polynôme $a_0+a_1X+\cdots+a_nX^n$ sous la forme d'une liste $[a_0,a_1,\ldots,a_n]$.

1. Écrire une fonction correspondant à la somme de deux polynômes. Calculer la complexité de cet algorithme (en terme du nombre d'additions sur les coefficients, en fonctions du degré des polynômes).
2. Écrire une fonction correspondant au produit de deux polynômes. Calculer la complexité de cet algorithme (en terme du nombre d'additions et de multiplications sur les coefficients).
3. Écrire une fonction correspondant au quotient et au reste de la division euclidienne de A par B où B est un polynôme unitaire (son coefficient de plus haut degré est 1). Majorer la complexité de cet algorithme (en terme du nombre d'additions et de multiplications sur les coefficients).

1. La seule difficulté est de gérer les indices, en particulier on ne peut appeler un élément d'une liste en dehors des indices où elle est définie. Une bonne idée consiste à commencer par définir une fonction `degre(poly)`, qui renvoie le degré du polynôme (attention au 0 non significatifs).
Voici le code dans le cas simple où $\deg A = \deg B$:

polynome.py (1)

```
def somme(A,B):            # si deg(A)=deg(B)
    C = []
    for i in range(0,degre(A)+1):
        s = A[i]+B[i]
        C.append(s)
```

Calculons sa complexité, on suppose $\deg A \leqslant n$ et $\deg B \leqslant n$: il faut faire l'addition des coefficients $a_i + b_i$, pour i variant de 0 à n : donc la complexité est de $n+1$ additions (dans $\mathbb{Z}$ ou $\mathbb{R}$).

2. Pour le produit il faut se rappeler que si $A(X) = \sum_{i=0}^{m} a_i X^i$, $B(X) = \sum_{j=0}^{n} b_j X^j$ et $C = A \times B = \sum_{k=0}^{m+n} c_k X^k$ alors le k-ème coefficient de C est $c_k = \sum_{i+j=k} a_i \times b_j$. Dans la pratique on fait attention de ne pas accéder à des coefficients qui n'ont pas été définis.

polynome.py (2)

```
def produit(A,B):
    C = []
    for k in range(degre(A)+degre(B)+1):
        s = 0
        for i in range(k+1):
            if (i <= degre(A)) and (k-i <= degre(B)):
                s = s + A[i]*B[k-i]
        C.append(s)
    return C
```

Pour la complexité on commence par compter le nombre de multiplications (dans $\mathbb{Z}$ ou $\mathbb{R}$). Notons $m = \deg A$ et $n = \deg B$. Alors il faut multiplier les $m+1$ coefficients de A par les $n+1$ coefficients de B : il y a donc $(m+1)(n+1)$ multiplications.

Comptons maintenant les additions : les coefficients de $A \times B$ sont : $c_0 = a_0 b_0$, $c_1 = a_0 b_1 + a_1 b_0$, $c_2 = a_2 b_0 + a_1 b_1 + a_2 b_0$,...

Nous utilisons l'astuce suivante : nous savons que le produit $A \times B$ est de degré $m+n$ donc a (au plus) $m+n+1$ coefficients. Partant de $(m+1)(n+1)$ produits, chaque addition regroupe deux termes, et nous devons arriver à $m+n+1$ coefficients. Il y a donc $(m+1)(n+1) - (m+n+1) = mn$ additions.

3. Pour la division euclidienne, le principe est de poser une division de polynôme. Par exemple pour $A = 2X^4 - X^3 - 2X^2 + 3X - 1$ et $B = X^2 - X + 1$.

$$
\begin{array}{rl|l}
 & 2X^4 - X^3 - 2X^2 + 3X - 1 & X^2 - X + 1 \\
- & 2X^4 - 2X^3 + 2X^2 & \hline \\
\hline
 & X^3 - 4X^2 + 3X - 1 & 2X^2 + X - 3 \\
- & X^3 - X^2 + X & \\
\hline
 & -3X^2 + 2X - 1 & \\
- & -3X^2 + 3X - 3 & \\
\hline
 & -X + 2 &
\end{array}
$$

Alors on cherche quel monôme P_1 fait diminuer le degré de $A - P_1 B$, c'est $2X^2$ (le coefficient 2 est le coefficient dominant de A). On pose ensuite $R_1 = A - P_1 B = X^3 - 4X^2 + 3X - 1$, $Q_1 = 2X^2$, on recommence avec R_1 divisé par B, $R_2 = R_1 - P_2 B$ avec $P_2 = X$, $Q_2 = Q_1 + P_2$,... On arrête lorsque $\deg R_i < \deg B$.

polynome.py (3)

```
def division(A,B):
    Q = [0]      # Quotient
    R = A        # Reste
    while (degre(R) >= degre(B)):
        P = monome(R[degre(R)],degre(R)-degre(B))
        R = somme(R,produit(-P,B))
        Q = somme(Q,P)
    return Q,R
```

C'est une version un peu simplifiée du code : où $P = r_n X^{\deg R - \deg B}$ et où il faut remplacer $-P$ par $[-a_0, -a_1, ...]$. Si $A, B \in \mathbb{Z}[X]$ alors le fait que B soit unitaire implique que Q et R sont aussi à coefficients entiers.

Quelle est la complexité de la division euclidienne ? À chaque étape on effectue une multiplication de polynômes $(P_i \times B)$ puis une addition de polynôme $(R_i - P_i B)$; à chaque étape le degré de R_i diminue (au moins) de 1. Donc il y a au plus $\deg A - \deg B + 1$ étapes.

Mais dans la pratique c'est plus simple que cela. La multiplication $P_i \times B$ est très simple : car P_i est un monôme $P_i = p_i X^i$. Multiplier par X^i c'est juste un décalage d'indice (comme multiplier par 10^i en écriture décimale) c'est donc une opération négligeable. Il reste donc à multiplier les coefficients de B par p_i : il y a donc $\deg B + 1$ multiplications de coefficients. La soustraction aussi est assez simple on retire à R_i un multiple de B, donc on a au plus $\deg B + 1$ coefficients à soustraire : il y a à chaque étape $\deg B + 1$ additions de coefficients.

Bilan : si $m = \deg A$ et $n = \deg B$ alors la division euclidienne s'effectue en au plus $(m-n+1)(m+1)$ multiplications et le même nombre d'additions (dans $\mathbb{Z}$ ou $\mathbb{R}$).

6.3 Algorithme de Karatsuba

Pour diminuer la complexité de la multiplication de polynômes, on va utiliser un paradigme très classique de programmation : « diviser pour régner ». Pour cela, on va décomposer les polynômes à multiplier P et Q de degrés strictement inférieurs à $2n$ en

$$P = P_1 + P_2 \cdot X^n \quad \text{et} \quad Q = Q_1 + Q_2 \cdot X^n$$

avec les degrés de P_1, P_2, Q_1 et Q_2 strictement inférieurs à n.

Travaux pratiques 22.

1. Écrire une formule qui réduit la multiplication des polynômes P et Q de degrés strictement inférieurs à $2n$ en multiplications de polynômes de degrés strictement inférieurs à n.
2. Programmer un algorithme récursif de multiplication qui utilise la formule précédente. Quelle est sa complexité ?
3. On peut raffiner cette méthode avec la remarque suivante de Karatsuba : le terme intermédiaire de $P \cdot Q$ s'écrit
$$P_1 \cdot Q_2 + P_2 \cdot Q_1 = (P_1 + P_2) \cdot (Q_1 + Q_2) - P_1 Q_1 - P_2 Q_2$$
Comme on a déjà calculé $P_1 Q_1$ et $P_2 Q_2$, on échange deux multiplications et une addition (à gauche) contre une multiplication et quatre additions (à droite). Écrire une fonction qui réalise la multiplication de polynômes à la Karatsuba.
4. Trouver la formule de récurrence qui définit la complexité de la multiplication de Karatsuba. Quelle est sa solution ?

1. Il suffit de développer le produit $(P_1 + X^n P_2) \cdot (Q_1 + X^n Q_2)$:
$$(P_1 + X^n P_2) \cdot (Q_1 + X^n Q_2) = P_1 Q_1 + X^n \cdot (P_1 Q_2 + P_2 Q_1) + X^{2n} \cdot P_2 Q_2$$

On se ramène ainsi aux quatre multiplications P_1Q_1 , P_1Q_2 , P_2Q_1 et P_2Q_2 entre polynômes de degrés strictement inférieurs à n, plus deux multiplications par X^n et X^{2n} qui ne sont que des ajouts de zéros en tête de liste.

2. On sépare les deux étapes de l'algorithme : d'abord la découpe des polynômes (dans laquelle il ne faut pas oublier de donner n en argument car ce n'est pas forcément le milieu du polynôme, n doit être le même pour P et Q). Le découpage `P1,P2 = decoupe(P,n)` correspond à l'écriture $P = P_1 + X^n P_2$.

polynome.py (4)

```
def decoupe(P,n):
    if (degre(P)<n): return P, [0]
    else: return P[0:n], P[n:]
```

On a aussi besoin d'une fonction `produit_monome(P,n)` qui renvoie le polynôme $X^n \cdot P$ par un décalage. Voici la multiplication proprement dite avec les appels récursifs et leur combinaison.

polynome.py (5)

```
def produit_assez_rapide(P,Q):
    p = degre(P) ; q = degre(Q)
    if (p == 0): return [P[0]*k for k in Q]  # Condition initiale: P=cst
    if (q == 0): return [Q[0]*k for k in P]  # Condition initiale: Q=cst
    n = (max(p,q)+1)//2                      # demi-degré
    P1,P2 = decoupe(P,n)                     # decoupages
    Q1,Q2 = decoupe(Q,n)
    P1Q1 = produit_assez_rapide(P1,Q1)       # produits en petits degrés
    P2Q2 = produit_assez_rapide(P2,Q2)
    P1Q2 = produit_assez_rapide(P1,Q2)
    P2Q1 = produit_assez_rapide(P2,Q1)
    R1 = produit_monome(somme(P1Q2,P2Q1),n)  # décalages
    R2 = produit_monome(P2Q2,2*n)
    return somme(P1Q1,somme(R1,R2))          # sommes
```

La relation de récurrence qui exprime la complexité de cet algorithme est $C(n) = 4C(n/2) + O(n)$ et elle se résout en $C(n) = O(n^2)$. Voir la question suivante pour une méthode de résolution.

3.

polynome.py (6)

```
def produit_rapide(P,Q):
    p = degre(P) ; q = degre(Q)
    if (p == 0): return [P[0]*k for k in Q]  # Condition initiale: P=cst
    if (q == 0): return [Q[0]*k for k in P]  # Condition initiale: Q=cst
    n = (max(p,q)+1)//2                      # demi-degré
    P1,P2 = decoupe(P,n)                     # decoupages
    Q1,Q2 = decoupe(Q,n)
    P1Q1 = produit_rapide(P1,Q1)             # produits en petits degrés
    P2Q2 = produit_rapide(P2,Q2)
    PQ = produit_rapide(somme(P1,P2),somme(Q1,Q2))
    R1 = somme(PQ,somme([-k for k in P1Q1],[-k for k in P2Q2]))
    R1 = produit_monome(R1,n)                # décalages
    R2 = produit_monome(P2Q2,2*n)
    return somme(P1Q1,somme(R1,R2))          # sommes
```

4. Notons $C(n)$ la complexité de la multiplication entre deux polynômes de degrés strictement inférieurs à n. En plus des trois appels récursifs, il y a des opérations linéaires : deux calculs de degrés, deux découpes en $n/2$ puis des additions : deux de taille $n/2$, une de taille n, une de taille $3n/2$ et une de taille $2n$. On obtient donc la relation de récurrence suivante :

$$C(n) = 3 \cdot C(n/2) + \gamma n$$

où $\gamma = \frac{15}{2}$. Une méthode de résolution est de poser $\alpha_\ell = \frac{C(2^\ell)}{3^\ell}$ qui vérifie $\alpha_\ell = \alpha_{\ell-1} + \gamma \left(\frac{2}{3}\right)^\ell$. D'où on tire, puisque $\alpha_0 = C(1) = 1$,

$$\alpha_\ell = \gamma \sum_{k=1}^{\ell} \left(\frac{2}{3}\right)^k + \alpha_0 = 3\gamma \left(1 - \left(\frac{2}{3}\right)^{\ell+1}\right) + 1 - \gamma$$

puis pour $n = 2^\ell$:

$$C(n) = C(2^\ell) = 3^\ell \alpha_\ell = \gamma(3^{\ell+1} - 2^{\ell+1}) + (1-\gamma)3^\ell = O(3^\ell) = O(2^{\ell \frac{\ln 3}{\ln 2}}) = O(n^{\frac{\ln 3}{\ln 2}})$$

La complexité de la multiplication de Karatsuba est donc $O(n^{\frac{\ln 3}{\ln 2}}) \simeq O(n^{1.585})$.

6.4 Optimiser ses algorithmes

Voici quelques petites astuces pour accélérer l'écriture ou la vitesse des algorithmes :
- `k ** 3` au lieu de `k * k * k` (cela économise de la mémoire, une seule variable au lieu de 3) ;
- `k ** 2 <= n` au lieu de `k <= sqrt(n)` (les calculs avec les entiers sont beaucoup plus rapides qu'avec les réels) ;
- `x += 1` au lieu de `x = x +1` (gain de mémoire) ;
- `a,b = a+b, a-b` au lieu de `newa = a+b ; newb = a-b ; a = newa ; b = newb` (gain de mémoire, code plus court).

Cependant il ne faut pas que cela nuise à la lisibilité du code : il est important que quelqu'un puisse relire et modifier votre code. Le plus souvent c'est vous même qui modifierez les algorithmes qui vous avez écrits et vous serez ravi d'y trouver des commentaires clairs et précis !

6.5 Mini-exercices

1. Faire une fonction qui renvoie le pgcd de deux polynômes.
2. Comparer les complexités des deux méthodes suivantes pour évaluer un polynôme P en une valeur $x_0 \in \mathbb{R}$: $P(x_0) = a_0 + a_1 x_0 + \cdots + a_{n-1} x_0^{n-1} + a_n x_0^n$ et $P(x_0) = a_0 + x_0\Big(a_1 + x_0\big(a_2 + \cdots + x_0(a_{n-1} + a_n x_0)\big)\Big)$ (méthode de Horner).
3. Comment trouver le maximum d'une liste ? Montrer que votre méthode est de complexité minimale (en terme du nombre de comparaisons).
4. Soit $f : [a,b] \to \mathbb{R}$ une fonction continue vérifiant $f(a) \cdot f(b) \leqslant 0$. Combien d'itérations de la méthode de dichotomie sont nécessaires pour obtenir une racine de $f(x) = 0$ avec une précision inférieure à ε ?
5. Programmer plusieurs façons de calculer les coefficients du binôme de Newton $\binom{n}{k}$ et les comparer.
6. Trouver une méthode de calcul de 2^n qui utilise peu de multiplications. On commencera par écrire n en base 2.

Auteurs
Rédaction : Arnaud Bodin
Relecture : Jean-François Barraud
Remerciements à Lionel Rieg pour son tp sur l'algorithme de Karatsuba

Deuxième partie

Les rappels de cours

Logique et raisonnements

Vidéo ■ partie 1. Logique
Vidéo ■ partie 2. Raisonnements
Fiche d'exercices ♦ Logique, ensembles, raisonnements

Quelques motivations

- Il est important d'avoir un ***langage rigoureux***. La langue française est souvent ambigüe. Prenons l'exemple de la conjonction « *ou* » ; au restaurant « *fromage ou dessert* » signifie l'un ou l'autre mais pas les deux. Par contre si dans un jeu de carte on cherche « *les as ou les cœurs* » alors il ne faut pas exclure l'as de cœur. Autre exemple : que répondre à la question « *As-tu* 10 *euros en poche ?* » si l'on dispose de 15 euros ?
- Il y a des notions difficiles à expliquer avec des mots : par exemple la continuité d'une fonction est souvent expliquée par « *on trace le graphe sans lever le crayon* ». Il est clair que c'est une définition peu satisfaisante. Voici la définition mathématique de la continuité d'une fonction $f : I \to \mathbb{R}$ en un point $x_0 \in I$:
$$\forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall x \in I \quad (|x - x_0| < \delta \implies |f(x) - f(x_0)| < \varepsilon).$$
C'est le but de ce chapitre de rendre cette ligne plus claire ! C'est la ***logique***.
- Enfin les mathématiques tentent de ***distinguer le vrai du faux***. Par exemple « *Est-ce qu'une augmentation de* 20%*, puis de* 30% *est plus intéressante qu'une augmentation de* 50% *?* ». Vous pouvez penser « *oui* » ou « *non* », mais pour en être sûr il faut suivre une démarche logique qui mène à la conclusion. Cette démarche doit être convaincante pour vous mais aussi pour les autres. On parle de ***raisonnement***.

Les mathématiques sont un langage pour s'exprimer rigoureusement, adapté aux phénomènes complexes, qui rend les calculs exacts et vérifiables. Le raisonnement est le moyen de valider — ou d'infirmer — une hypothèse et de l'expliquer à autrui.

1 Logique

1.1 Assertions

Une ***assertion*** est une phrase soit vraie, soit fausse, pas les deux en même temps.
Exemples :
- « *Il pleut.* »
- « *Je suis plus grand que toi.* »
- « $2+2=4$ »
- « $2\times 3=7$ »
- « *Pour tout* $x\in\mathbb{R}$, *on a* $x^2\geqslant 0$. »
- « *Pour tout* $z\in\mathbb{C}$, *on a* $|z|=1$. »

Si P est une assertion et Q est une autre assertion, nous allons définir de nouvelles assertions construites à partir de P et de Q.

L'opérateur logique « *et* »

L'assertion « P **et** Q » est vraie si P est vraie et Q est vraie. L'assertion « P *et* Q » est fausse sinon.
On résume ceci en une ***table de vérité*** :

$P \setminus Q$	V	F
V	V	F
F	F	F

FIGURE 4.1 – Table de vérité de « P *et* Q »

Par exemple si P est l'assertion « *Cette carte est un as* » et Q l'assertion « *Cette carte est cœur* » alors l'assertion « P *et* Q » est vraie si la carte est l'as de cœur et est fausse pour toute autre carte.

L'opérateur logique « *ou* »

L'assertion « P **ou** Q » est vraie si l'une des deux assertions P ou Q est vraie. L'assertion « P *ou* Q » est fausse si les deux assertions P et Q sont fausses.
On reprend ceci dans la table de vérité :

$P \setminus Q$	V	F
V	V	V
F	V	F

FIGURE 4.2 – Table de vérité de « P *ou* Q »

Si P est l'assertion « *Cette carte est un as* » et Q l'assertion « *Cette carte est cœur* » alors l'assertion « P *ou* Q » est vraie si la carte est un as ou bien un cœur (en particulier elle est vraie pour l'as de cœur).

Remarque. Pour définir les opérateurs « *ou* », « *et* » on fait appel à une phrase en français utilisant les mots *ou*, *et* ! Les tables de vérités permettent d'éviter ce problème.

La négation « *non* »

L'assertion « **non** P » est vraie si P est fausse, et fausse si P est vraie.

P	V	F
non P	F	V

FIGURE 4.3 – Table de vérité de « *non* P »

L'implication $\Longrightarrow$

La définition mathématique est la suivante :

L'assertion « *(non P) ou Q* » est notée « $P \Longrightarrow Q$ ».

Sa table de vérité est donc la suivante :

$P \setminus Q$	V	F
V	V	F
F	V	V

FIGURE 4.4 – Table de vérité de « $P \Longrightarrow Q$ »

L'assertion « $P \Longrightarrow Q$ » se lit en français « *P implique Q* ».
Elle se lit souvent aussi « *si P est vraie alors Q est vraie* » ou « *si P alors Q* ».
Par exemple :
- « $0 \leqslant x \leqslant 25 \Longrightarrow \sqrt{x} \leqslant 5$ » est vraie (prendre la racine carrée).
- « $x \in]-\infty, -4[\Longrightarrow x^2 + 3x - 4 > 0$ » est vraie (étudier le binôme).
- « $\sin(\theta) = 0 \Longrightarrow \theta = 0$ » est fausse (regarder pour $\theta = 2\pi$ par exemple).
- « $2+2=5 \Longrightarrow \sqrt{2} = 2$ » est vraie ! Eh oui, si P est fausse alors l'assertion « $P \Longrightarrow Q$ » est toujours vraie.

L'équivalence $\Longleftrightarrow$

L'***équivalence*** est définie par :

« $P \Longleftrightarrow Q$ » est l'assertion « *(P* $\Longrightarrow$ *Q) et (Q* $\Longrightarrow$ *P)* ».

On dira « *P est équivalent à Q* » ou « *P équivaut à Q* » ou « *P si et seulement si Q* ». Cette assertion est vraie lorsque P et Q sont vraies ou lorsque P et Q sont fausses. La table de vérité est :

$P \setminus Q$	V	F
V	V	F
F	F	V

FIGURE 4.5 – Table de vérité de « $P \Longleftrightarrow Q$ »

Exemples :
- Pour $x, x' \in \mathbb{R}$, l'équivalence « $x \cdot x' = 0 \Longleftrightarrow (x = 0$ *ou* $x' = 0)$ » est vraie.
- Voici une équivalence *toujours fausse* (quelque soit l'assertion P) : « $P \Longleftrightarrow non(P)$ ».

On s'intéresse davantage aux assertions vraies qu'aux fausses, aussi dans la pratique et en dehors de ce chapitre on écrira « $P \Longleftrightarrow Q$ » ou « $P \Longrightarrow Q$ » uniquement lorsque ce sont des assertions vraies. Par exemple si l'on écrit « $P \Longleftrightarrow Q$ » cela sous-entend « $P \Longleftrightarrow Q$ *est vraie* ». Attention rien ne dit que P et Q soient vraies. Cela signifie que P et Q sont vraies en même temps ou fausses en même temps.

Proposition 9.
Soient P, Q, R trois assertions. Nous avons les équivalences (vraies) suivantes :

1. $P \Longleftrightarrow non(non(P))$
2. $(P$ *et* $Q) \Longleftrightarrow (Q$ *et* $P)$
3. $(P$ *ou* $Q) \Longleftrightarrow (Q$ *ou* $P)$
4. $non(P$ *et* $Q) \Longleftrightarrow (non\ P)$ *ou* $(non\ Q)$
5. $non(P$ *ou* $Q) \Longleftrightarrow (non\ P)$ *et* $(non\ Q)$
6. $\big(P$ *et* $(Q$ *ou* $R)\big) \Longleftrightarrow (P$ *et* $Q)$ *ou* $(P$ *et* $R)$
7. $\big(P$ *ou* $(Q$ *et* $R)\big) \Longleftrightarrow (P$ *ou* $Q)$ *et* $(P$ *ou* $R)$

8. « $P \implies Q$ » $\iff$ « $non(Q) \implies non(P)$ »

Démonstration. Voici des exemples de démonstrations :

4. Il suffit de comparer les deux assertions « $non(P \text{ et } Q)$ » et « $(non\ P)$ *ou* $(non\ Q)$ » pour toutes les valeurs possibles de P et Q. Par exemple si P est vrai et Q est vrai alors « P *et* Q » est vrai donc « $non(P \text{ et } Q)$ » est faux ; d'autre part (non P) est faux, (non Q) est faux donc « $(non\ P)$ *ou* $(non\ Q)$ » est faux. Ainsi dans ce premier cas les assertions sont toutes les deux fausses. On dresse ainsi les deux tables de vérités et comme elles sont égales les deux assertions sont équivalentes.

$P \setminus Q$	V	F
V	F	V
F	V	V

FIGURE 4.6 – Tables de vérité de « $non(P \text{ et } Q)$ » et de « $(non\ P)$ *ou* $(non\ Q)$ »

6. On fait la même chose mais il y a trois variables : P, Q, R. On compare donc les tables de vérité d'abord dans le cas où P est vrai (à gauche), puis dans le cas où P est faux (à droite). Dans les deux cas les deux assertions « $\big(P \text{ et } (Q \text{ ou } R)\big)$ » et « $(P \text{ et } Q)$ *ou* $(P \text{ et } R)$ » ont la même table de vérité donc les assertions sont équivalentes.

$Q \setminus R$	V	F
V	V	V
F	V	F

$Q \setminus R$	V	F
V	F	F
F	F	F

8. Par définition, l'implication « $P \implies Q$ » est l'assertion « *(non P) ou* Q ».
Donc l'implication « $non(Q) \implies non(P)$ » est équivalente à « $non(non(Q))$ *ou* $non(P)$ » qui équivaut encore à « Q *ou* $non(P)$ » et donc est équivalente à « $P \implies Q$ ». On aurait aussi pu encore une fois dresser les deux tables de vérité et voir quelles sont égales.

□

1.2 Quantificateurs

Le quantificateur ∀ : « *pour tout* »

Une assertion P peut dépendre d'un paramètre x, par exemple « $x^2 \geqslant 1$ », l'assertion $P(x)$ est vraie ou fausse selon la valeur de x.
L'assertion

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

est une assertion vraie lorsque les assertions $P(x)$ sont vraies pour tous les éléments x de l'ensemble E. On lit « *Pour tout x appartenant à E,* $P(x)$ », sous-entendu « *Pour tout x appartenant à E,* $P(x)$ *est vraie* ».
Par exemple :
– « $\forall x \in [1, +\infty[\quad (x^2 \geqslant 1)$ » est une assertion vraie.
– « $\forall x \in \mathbb{R} \quad (x^2 \geqslant 1)$ » est une assertion fausse.
– « $\forall n \in \mathbb{N} \quad n(n+1)$ *est divisible par* 2 » est vraie.

Le quantificateur ∃ : « *il existe* »

L'assertion

$$\exists x \in E \quad P(x)$$

est une assertion vraie lorsque l'on peut trouver au moins un x de E pour lequel $P(x)$ est vraie. On lit « *il existe x appartenant à E tel que* $P(x)$ *(soit vraie)* ».
Par exemple :
– « $\exists x \in \mathbb{R} \quad (x(x-1) < 0)$ » est vraie (par exemple $x = \frac{1}{2}$ vérifie bien la propriété).
– « $\exists n \in \mathbb{N} \quad n^2 - n > n$ » est vraie (il y a plein de choix, par exemple $n = 3$ convient, mais aussi $n = 10$ ou même $n = 100$, un seul suffit pour dire que l'assertion est vraie).
– « $\exists x \in \mathbb{R} \quad (x^2 = -1)$ » est fausse (aucun réel au carré ne donnera un nombre négatif).

La négation des quantificateurs

La négation de « $\forall x \in E \quad P(x)$ » est « $\exists x \in E \quad non\ P(x)$ » .

Par exemple la négation de « $\forall x \in [1, +\infty[\quad (x^2 \geqslant 1)$ » est l'assertion « $\exists x \in [1, +\infty[\quad (x^2 < 1)$ ». En effet la négation de $x^2 \geqslant 1$ est non$(x^2 \geqslant 1)$ mais s'écrit plus simplement $x^2 < 1$.

La négation de « $\exists x \in E \quad P(x)$ » est « $\forall x \in E \quad non\ P(x)$ ».

Voici des exemples :
- La négation de « $\exists z \in \mathbb{C} \quad (z^2 + z + 1 = 0)$ » est « $\forall z \in \mathbb{C} \quad (z^2 + z + 1 \neq 0)$ ».
- La négation de « $\forall x \in \mathbb{R} \quad (x + 1 \in \mathbb{Z})$ » est « $\exists x \in \mathbb{R} \quad (x + 1 \notin \mathbb{Z})$ ».
- Ce n'est pas plus difficile d'écrire la négation de phrases complexes. Pour l'assertion :

$$\forall x \in \mathbb{R} \quad \exists y > 0 \quad (x + y > 10)$$

sa négation est

$$\exists x \in \mathbb{R} \quad \forall y > 0 \quad (x + y \leqslant 10).$$

Remarques

L'ordre des quantificateurs est très important. Par exemple les deux phrases logiques

$$\forall x \in \mathbb{R} \quad \exists y \in \mathbb{R} \quad (x + y > 0) \qquad \text{et} \qquad \exists y \in \mathbb{R} \quad \forall x \in \mathbb{R} \quad (x + y > 0).$$

sont différentes. La première est vraie, la seconde est fausse. En effet une phrase logique se lit de gauche à droite, ainsi la première phrase affirme « *Pour tout réel x, il existe un réel y (qui peut donc dépendre de x) tel que* $x + y > 0$. » (par exemple on peut prendre $y = x + 1$). C'est donc une phrase vraie. Par contre la deuxième se lit : « *Il existe un réel y, tel que pour tout réel x,* $x + y > 0$. » Cette phrase est fausse, cela ne peut pas être le même y qui convient pour tous les x !
On retrouve la même différence dans les phrases en français suivantes. Voici une phrase vraie « *Pour toute personne, il existe un numéro de téléphone* », bien sûr le numéro dépend de la personne. Par contre cette phrase est fausse : « *Il existe un numéro, pour toutes les personnes* ». Ce serait le même numéro pour tout le monde !

Terminons avec d'autres remarques.
- Quand on écrit « $\exists x \in \mathbb{R} \quad (f(x) = 0)$ » cela signifie juste qu'il existe un réel pour lequel f s'annule. Rien ne dit que ce x est unique. Dans un premier temps vous pouvez lire la phrase ainsi : « *il existe* au moins *un réel x tel que* $f(x) = 0$ ». Afin de préciser que f s'annule en une unique valeur, on rajoute un point d'exclamation :
$$\exists ! x \in \mathbb{R} \quad (f(x) = 0).$$
- Pour la négation d'une phrase logique, il n'est pas nécessaire de savoir si la phrase est fausse ou vraie. Le procédé est algorithmique : on change le « *pour tout* » en « *il existe* » et inversement, puis on prend la négation de l'assertion P.
- Pour la négation d'une proposition, il faut être précis : la négation de l'inégalité stricte « $<$ » est l'inégalité large « $\geqslant$ », et inversement.
- Les quantificateurs ne sont pas des abréviations. Soit vous écrivez une phrase en français : « *Pour tout réel x, si* $f(x) = 1$ *alors* $x \geqslant 0$. » , soit vous écrivez la phrase logique :
$$\forall x \in \mathbb{R} \quad (f(x) = 1 \implies x \geqslant 0).$$
Mais surtout n'écrivez pas « $\forall x$ *réel, si* $f(x) = 1 \implies x$ *positif ou nul* ». Enfin, pour passer d'une ligne à l'autre d'un raisonnement, préférez plutôt « *donc* » à « $\implies$ ».
- Il est défendu d'écrire $\nexists, \not\Longrightarrow$. Ces symboles n'existent pas !

1.3 Mini-exercices

1. Écrire la table de vérité du « *ou exclusif* ». (C'est le *ou* dans la phrase « *fromage ou dessert* », l'un ou l'autre mais pas les deux.)
2. Écrire la table de vérité de « *non (P et Q)* ». Que remarquez vous ?
3. Écrire la négation de « $P \implies Q$ ».
4. Démontrer les assertions restantes de la proposition 9.
5. Écrire la négation de « $\big(P$ *et* $(Q$ *ou* $R)\big)$ ».
6. Écrire à l'aide des quantificateurs la phrase suivante : « *Pour tout nombre réel, son carré est positif* ». Puis écrire la négation.
7. Mêmes questions avec les phrases : « *Pour chaque réel, je peux trouver un entier relatif tel que leur produit soit strictement plus grand que* 1 ». Puis « *Pour tout entier n, il existe un unique réel x tel que* $\exp(x)$ *égale* n ».

2 Raisonnements

Voici des méthodes classiques de raisonnements.

2.1 Raisonnement direct

On veut montrer que l'assertion « $P \implies Q$ » est vraie. On suppose que P est vraie et on montre qu'alors Q est vraie. C'est la méthode à laquelle vous êtes le plus habitué.

Exemple 19. Montrer que si $a, b \in \mathbb{Q}$ alors $a+b \in \mathbb{Q}$.

Démonstration. Prenons $a \in \mathbb{Q}$, $b \in \mathbb{Q}$. Rappelons que les rationnels $\mathbb{Q}$ sont l'ensemble des réels s'écrivant $\frac{p}{q}$ avec $p \in \mathbb{Z}$ et $q \in \mathbb{N}^*$.
Alors $a = \frac{p}{q}$ pour un certain $p \in \mathbb{Z}$ et un certain $q \in \mathbb{N}^*$. De même $b = \frac{p'}{q'}$ avec $p' \in \mathbb{Z}$ et $q' \in \mathbb{N}^*$. Maintenant

$$a+b = \frac{p}{q} + \frac{p'}{q'} = \frac{pq'+qp'}{qq'}.$$

Or le numérateur $pq'+qp'$ est bien un élément de $\mathbb{Z}$; le dénominateur qq' est lui un élément de $\mathbb{N}^*$. Donc $a+b$ s'écrit bien de la forme $a+b = \frac{p''}{q''}$ avec $p'' \in \mathbb{Z}$, $q'' \in \mathbb{N}^*$. Ainsi $a+b \in \mathbb{Q}$. □

2.2 Cas par cas

Si l'on souhaite vérifier une assertion $P(x)$ pour tous les x dans un ensemble E, on montre l'assertion pour les x dans une partie A de E, puis pour les x n'appartenant pas à A. C'est la méthode de ***disjonction*** ou du ***cas par cas***.

Exemple 20. Montrer que pour tout $x \in \mathbb{R}$, $|x-1| \leqslant x^2 - x + 1$.

Démonstration. Soit $x \in \mathbb{R}$. Nous distinguons deux cas.
Premier cas : $x \geqslant 1$**.** Alors $|x-1| = x-1$. Calculons alors $x^2 - x + 1 - |x-1|$.

$$\begin{aligned} x^2 - x + 1 - |x-1| &= x^2 - x + 1 - (x-1) \\ &= x^2 - 2x + 2 \\ &= (x-1)^2 + 1 \geqslant 0. \end{aligned}$$

Ainsi $x^2 - x + 1 - |x-1| \geqslant 0$ et donc $x^2 - x + 1 \geqslant |x-1|$.
Deuxième cas : $x < 1$**.** Alors $|x-1| = -(x-1)$. Nous obtenons $x^2 - x + 1 - |x-1| = x^2 - x + 1 + (x-1) = x^2 \geqslant 0$. Et donc $x^2 - x + 1 \geqslant |x-1|$.
Conclusion. Dans tous les cas $|x-1| \leqslant x^2 - x + 1$. □

2.3 Contraposée

Le raisonnement par ***contraposition*** est basé sur l'équivalence suivante (voir la proposition 9) :

L'assertion « $P \implies Q$ » est équivalente à « $non(Q) \implies non(P)$ ».

Donc si l'on souhaite montrer l'assertion « $P \implies Q$ », on montre en fait que si non(Q) est vraie alors non(P) est vraie.

Exemple 21. Soit $n \in \mathbb{N}$. Montrer que si n^2 est pair alors n est pair.

Démonstration. Nous supposons que n n'est pas pair. Nous voulons montrer qu'alors n^2 n'est pas pair. Comme n n'est pas pair, il est impair et donc il existe $k \in \mathbb{N}$ tel que $n = 2k+1$. Alors $n^2 = (2k+1)^2 = 4k^2+4k+1 = 2\ell+1$ avec $\ell = 2k^2+2k \in \mathbb{N}$. Et donc n^2 est impair.
Conclusion : nous avons montré que si n est impair alors n^2 est impair. Par contraposition ceci est équivalent à : si n^2 est pair alors n est pair. □

2.4 Absurde

Le ***raisonnement par l'absurde*** pour montrer « $P \implies Q$ » repose sur le principe suivant : on suppose à la fois que P est vraie et que Q est fausse et on cherche une contradiction. Ainsi si P est vraie alors Q doit être vraie et donc « $P \implies Q$ » est vraie.

Exemple 22. Soient $a, b \geqslant 0$. Montrer que si $\frac{a}{1+b} = \frac{b}{1+a}$ alors $a = b$.

Démonstration. Nous raisonnons par l'absurde en supposant que $\frac{a}{1+b} = \frac{b}{1+a}$ **et** $a \neq b$. Comme $\frac{a}{1+b} = \frac{b}{1+a}$ alors $a(1+a) = b(1+b)$ donc $a+a^2 = b+b^2$ d'où $a^2-b^2 = b-a$. Cela conduit à $(a-b)(a+b) = -(a-b)$. Comme $a \neq b$ alors $a-b \neq 0$ et donc en divisant par $a-b$ on obtient $a+b = -1$. La somme de deux nombres positifs ne peut être négative. Nous obtenons une contradiction.
Conclusion : si $\frac{a}{1+b} = \frac{b}{1+a}$ alors $a = b$. □

Dans la pratique, on peut choisir indifféremment entre un raisonnement par contraposition ou par l'absurde. Attention cependant de bien écrire quel type de raisonnement vous choisissez et surtout de ne pas changer en cours de rédaction !

2.5 Contre-exemple

Si l'on veut montrer qu'une assertion du type « $\forall x \in E \quad P(x)$ » est vraie alors pour chaque x de E il faut montrer que $P(x)$ est vraie. Par contre pour montrer que cette assertion est fausse alors il suffit de trouver $x \in E$ tel que $P(x)$ soit fausse. (Rappelez-vous la négation de « $\forall x \in E \quad P(x)$ » est « $\exists x \in E \quad non\ P(x)$ »). Trouver un tel x c'est trouver un ***contre-exemple*** à l'assertion « $\forall x \in E \quad P(x)$ ».

Exemple 23. Montrer que l'assertion suivante est fausse « *Tout entier positif est somme de trois carrés* ». (Les carrés sont les 0^2, 1^2, 2^2, 3^2,... Par exemple $6 = 2^2+1^2+1^2$.)

Démonstration. Un contre-exemple est 7 : les carrés inférieurs à 7 sont 0, 1, 4 mais avec trois de ces nombres on ne peut faire 7. □

2.6 Récurrence

Le ***principe de récurrence*** permet de montrer qu'une assertion $P(n)$, dépendant de n, est vraie pour tout $n \in \mathbb{N}$. La démonstration par récurrence se déroule en trois étapes : lors de l'***initialisation*** on prouve $P(0)$. Pour l'étape d'***hérédité***, on suppose $n \geqslant 0$ donné avec $P(n)$ vraie, et on démontre alors que l'assertion $P(n+1)$ au rang suivant est vraie. Enfin dans la ***conclusion***, on rappelle que par le principe de récurrence $P(n)$ est vraie pour tout $n \in \mathbb{N}$.

Exemple 24. Montrer que pour tout $n \in \mathbb{N}$, $2^n > n$.

Démonstration. Pour $n \geqslant 0$, notons $P(n)$ l'assertion suivante :

$$2^n > n.$$

Nous allons démontrer par récurrence que $P(n)$ est vraie pour tout $n \geqslant 0$.
Initialisation. Pour $n = 0$ nous avons $2^0 = 1 > 0$. Donc $P(0)$ est vraie.
Hérédité. Fixons $n \geqslant 0$. Supposons que $P(n)$ soit vraie. Nous allons montrer que $P(n+1)$ est vraie.

$$\begin{aligned} 2^{n+1} &= 2^n + 2^n \\ &> n + 2^n \qquad \text{car par } P(n) \text{ nous savons } 2^n > n, \\ &> n + 1 \qquad \text{car } 2^n \geqslant 1. \end{aligned}$$

Donc $P(n+1)$ est vraie.
Conclusion. Par le principe de récurrence $P(n)$ est vraie pour tout $n \geqslant 0$, c'est-à-dire $2^n > n$ pour tout $n \geqslant 0$. □

Remarques :
- La rédaction d'une récurrence est assez rigide. Respectez scrupuleusement la rédaction proposée : donnez un nom à l'assertion que vous souhaitez montrer (ici $P(n)$), respectez les trois étapes (même si souvent l'étape d'initialisation est très facile). En particulier méditez et conservez la première ligne de l'hérédité « Fixons $n \geqslant 0$. Supposons que $P(n)$ soit vraie. Nous allons montrer que $P(n+1)$ est vraie. »
- Si on doit démontrer qu'une propriété est vraie pour tout $n \geqslant n_0$, alors on commence l'initialisation au rang n_0.
- Le principe de récurrence est basé sur la construction de $\mathbb{N}$. En effet un des axiomes pour définir $\mathbb{N}$ est le suivant : « *Soit A une partie de $\mathbb{N}$ qui contient 0 et telle que si $n \in A$ alors $n+1 \in A$. Alors $A = \mathbb{N}$* ».

2.7 Mini-exercices

1. (Raisonnement direct) Soient $a, b \in \mathbb{R}_+$. Montrer que si $a \leqslant b$ alors $a \leqslant \frac{a+b}{2} \leqslant b$ et $a \leqslant \sqrt{ab} \leqslant b$.
2. (Cas par cas) Montrer que pour tout $n \in \mathbb{N}$, $n(n+1)$ est divisible par 2 (distinguer les n pairs des n impairs).
3. (Contraposée ou absurde) Soient $a, b \in \mathbb{Z}$. Montrer que si $b \neq 0$ alors $a + b\sqrt{2} \notin \mathbb{Q}$. (On utilisera que $\sqrt{2} \notin \mathbb{Q}$.)
4. (Absurde) Soit $n \in \mathbb{N}^*$. Montrer que $\sqrt{n^2+1}$ n'est pas un entier.
5. (Contre-exemple) Est-ce que pour tout $x \in \mathbb{R}$ on a $x < 2 \implies x^2 < 4$?
6. (Récurrence) Montrer que pour tout $n \geqslant 1$, $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$.
7. (Récurrence) Fixons un réel $x \geqslant 0$. Montrer que pour tout entier $n \geqslant 1$, $(1+x)^n \geqslant 1 + nx$.

Auteurs
Arnaud Bodin
Benjamin Boutin
Pascal Romon

Ensembles et applications

Vidéo ■ partie 1. Ensembles
Vidéo ■ partie 2. Applications
Vidéo ■ partie 3. Injection, surjection, bijection
Vidéo ■ partie 4. Ensembles finis
Vidéo ■ partie 5. Relation d'équivalence
Fiche d'exercices ♦ Logique, ensembles, raisonnements
Fiche d'exercices ♦ Injection, surjection, bijection
Fiche d'exercices ♦ Dénombrement
Fiche d'exercices ♦ Relation d'équivalence, relation d'ordre

Motivations

Au début du XX^{e} siècle le professeur Frege peaufinait la rédaction du second tome d'un ouvrage qui souhaitait refonder les mathématiques sur des bases logiques. Il reçut une lettre d'un tout jeune mathématicien : *« J'ai bien lu votre premier livre. Malheureusement vous supposez qu'il existe un ensemble qui contient tous les ensembles. Un tel ensemble ne peut exister. »* S'ensuit une démonstration de deux lignes. Tout le travail de Frege s'écroulait et il ne s'en remettra jamais. Le jeune Russell deviendra l'un des plus grands logiciens et philosophes de sont temps. Il obtient le prix Nobel de littérature en 1950.
Voici le « paradoxe de Russell » pour montrer que l'ensemble de tous les ensembles ne peut exister. C'est très bref, mais difficile à appréhender. Par l'absurde, supposons qu'un tel ensemble $\mathscr{E}$ contenant tous les ensembles existe. Considérons

$$F = \Big\{E \in \mathscr{E} \mid E \notin E\Big\}.$$

Expliquons l'écriture $E \notin E$: le E de gauche est considéré comme un élément, en effet l'ensemble $\mathscr{E}$ est l'ensemble de tous les ensembles et E est un élément de cet ensemble ; le E de droite est considéré comme un ensemble, en effet les élément de $\mathscr{E}$ sont des ensembles ! On peut donc s'interroger si l'élément E appartient à l'ensemble E. Si non, alors par définition on met E dans l'ensemble F.
La contradiction arrive lorsque l'on se pose la question suivante : a-t-on $F \in F$ ou $F \notin F$? L'une des deux affirmation doit être vraie. Et pourtant :
– Si $F \in F$ alors par définition de F, F est l'un des ensembles E tel que $F \notin F$. Ce qui est contradictoire.
– Si $F \notin F$ alors F vérifie bien la propriété définissant F donc $F \in F$! Encore contradictoire.
Aucun des cas n'est possible. On en déduit qu'il ne peut exister un tel ensemble $\mathscr{E}$ contenant tous les ensembles.
Ce paradoxe a été popularisé par l'énigme suivante : *« Dans une ville, le barbier rase tous ceux qui ne se rasent pas eux-mêmes. Qui rase le barbier ? »* La seule réponse valable est qu'une telle situation ne peut exister.

Ne vous inquiétez pas, Russell et d'autres ont fondé la logique et les ensembles sur des bases solides. Cependant il n'est pas possible dans ce cours de tout redéfinir. Heureusement, vous connaissez déjà quelques ensembles :
– l'ensemble des entiers naturels $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
– l'ensemble des entiers relatifs $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
– l'ensemble des rationnels $\mathbb{Q} = \left\{\frac{p}{q} \mid p \in \mathbb{Z}, q \in \mathbb{N} \setminus \{0\}\right\}$.
– l'ensemble des réels $\mathbb{R}$, par exemple $1, \sqrt{2}, \pi, \ln(2), \ldots$
– l'ensemble des nombres complexes $\mathbb{C}$.

Nous allons essayer de voir les propriétés des ensembles, sans s'attacher à un exemple particulier. Vous vous apercevrez assez rapidement que ce qui est au moins aussi important que les ensembles, ce sont les relations entre ensembles : ce sera la notion d'application (ou fonction) entre deux ensembles.

1 Ensembles

1.1 Définir des ensembles

– On va définir informellement ce qu'est un ensemble : un ***ensemble*** est une collection d'éléments.
– Exemples :

$$\{0, 1\}, \quad \{\text{rouge}, \text{noir}\}, \quad \{0, 1, 2, 3, \ldots\} = \mathbb{N}.$$

– Un ensemble particulier est l'***ensemble vide***, noté $\varnothing$ qui est l'ensemble ne contenant aucun élément.
– On note

$$\boxed{x \in E}$$

si x est un élément de E, et $x \notin E$ dans le cas contraire.
– Voici une autre façon de définir des ensembles : une collection d'éléments qui vérifient une propriété.
– Exemples :

$$\{x \in \mathbb{R} \mid |x - 2| < 1\}, \quad \{z \in \mathbb{C} \mid z^5 = 1\}, \quad \{x \in \mathbb{R} \mid 0 \leqslant x \leqslant 1\} = [0, 1].$$

1.2 Inclusion, union, intersection, complémentaire

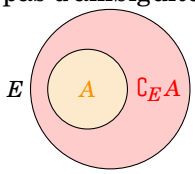
- L'***inclusion***. $E \subset F$ si tout élément de E est aussi un élément de F (autrement dit : $\forall x \in E\ (x \in F)$). On dit alors que E est un ***sous-ensemble*** de F ou une ***partie*** de F.
- L'***égalité***. $E = F$ si et seulement si $E \subset F$ et $F \subset E$.
- ***Ensemble des parties*** de E. On note $\mathscr{P}(E)$ l'ensemble des parties de E. Par exemple si $E = \{1,2,3\}$:

$$\mathscr{P}(\{1,2,3\}) = \big\{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\big\}.$$

- ***Complémentaire***. Si $A \subset E$,

$$\boxed{\complement_E A = \{x \in E \mid x \notin A\}}$$

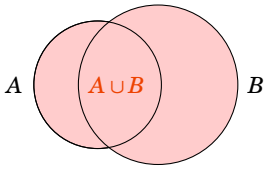
On le note aussi $E \setminus A$ et juste $\complement A$ s'il n'y a pas d'ambiguïté (et parfois aussi A^c ou $\overline{A}$).



- ***Union***. Pour $A, B \subset E$,

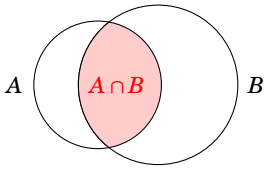
$$\boxed{A \cup B = \{x \in E \mid x \in A \text{ ou } x \in B\}}$$

Le «*ou*» n'est pas exclusif : x peut appartenir à A et à B en même temps.



- ***Intersection***.

$$\boxed{A \cap B = \{x \in E \mid x \in A \text{ et } x \in B\}}$$



1.3 Règles de calculs

Soient A, B, C des parties d'un ensemble E.

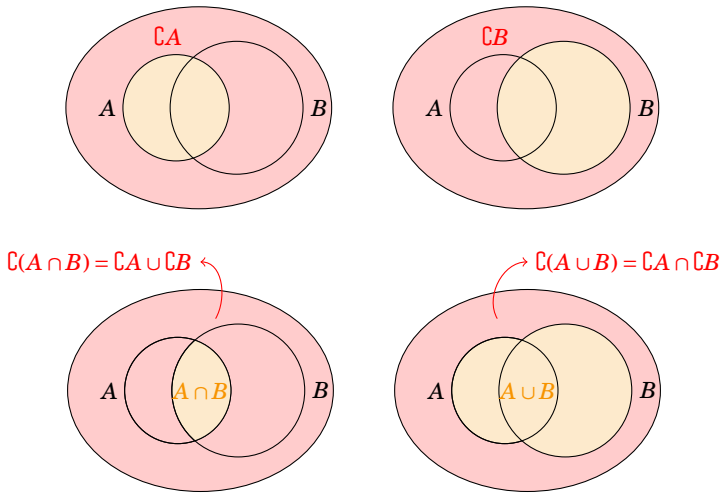
- $A \cap B = B \cap A$
- $A \cap (B \cap C) = (A \cap B) \cap C$ (on peut donc écrire $A \cap B \cap C$ sans ambigüité)
- $A \cap \varnothing = \varnothing$, $A \cap A = A$, $A \subset B \iff A \cap B = A$

- $A \cup B = B \cup A$
- $A \cup (B \cup C) = (A \cup B) \cup C$ (on peut donc écrire $A \cup B \cup C$ sans ambiguïté)
- $A \cup \varnothing = A$, $A \cup A = A$, $A \subset B \iff A \cup B = B$

- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

- $\complement(\complement A) = A$ et donc $A \subset B \iff \complement B \subset \complement A$.
- $\complement(A \cap B) = \complement A \cup \complement B$
- $\complement(A \cup B) = \complement A \cap \complement B$

Voici les dessins pour les deux dernières assertions.



Les preuves sont pour l'essentiel une reformulation des opérateurs logiques, en voici quelques-unes :
- Preuve de $A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$: $x\in A\cap(B\cup C) \iff x\in A$ et $x\in(B\cup C) \iff x\in A$ et ($x\in B$ ou $x\in C$) $\iff$ ($x\in A$ et $x\in B$) ou ($x\in A$ et $x\in C$) $\iff$ ($x\in A\cap B$) ou ($x\in A\cap C$) $\iff x\in(A\cap B)\cup(A\cap C)$.
- Preuve de $\complement(A\cap B)=\complement A\cup\complement B$: $x\in\complement(A\cap B) \iff x\notin(A\cap B) \iff \text{non}\big(x\in A\cap B\big) \iff \text{non}\big(x\in A$ et $x\in B\big) \iff \text{non}(x\in A)$ ou $\text{non}(x\in B) \iff x\notin A$ ou $x\notin B \iff x\in\complement A\cup\complement B$.

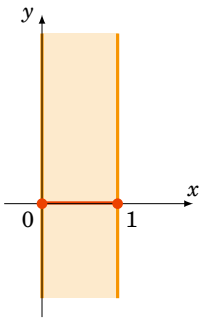
Remarquez que l'on repasse aux éléments pour les preuves.

1.4 Produit cartésien

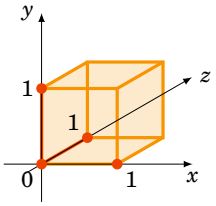
Soient E et F deux ensembles. Le ***produit cartésien***, noté $E\times F$, est l'ensemble des couples (x,y) où $x\in E$ et $y\in F$.

Exemple 25.

1. Vous connaissez $\mathbb{R}^2=\mathbb{R}\times\mathbb{R}=\{(x,y)\mid x,y\in\mathbb{R}\}$.
2. Autre exemple $[0,1]\times\mathbb{R}=\{(x,y)\mid 0\leqslant x\leqslant 1, y\in\mathbb{R}\}$



3. $[0,1]\times[0,1]\times[0,1]=\{(x,y,z)\mid 0\leqslant x,y,z\leqslant 1\}$



1.5 Mini-exercices

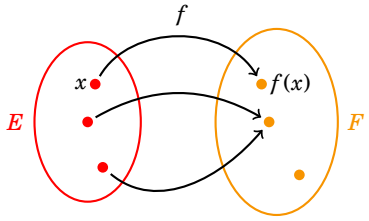
1. En utilisant les définitions, montrer : $A\neq B$ si et seulement s'il existe $a\in A\setminus B$ ou $b\in B\setminus A$.
2. Énumérer $\mathscr{P}(\{1,2,3,4\})$.
3. Montrer $A\cup(B\cap C)=(A\cup B)\cap(A\cup C)$ et $\complement(A\cup B)=\complement A\cap\complement B$.

4. Énumérer $\{1,2,3\}\times\{1,2,3,4\}$.
5. Représenter les sous-ensembles de $\mathbb{R}^2$ suivants : $\big(]0,1[\cup[2,3[\big)\times[-1,1]$, $\big(\mathbb{R}\setminus(]0,1[\cup[2,3[\big)\times\big((\mathbb{R}\setminus[-1,1])\cap[0,2]\big)$.

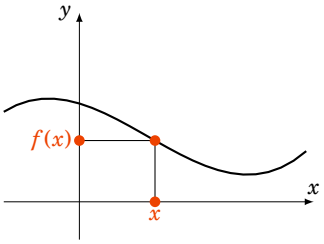
2 Applications

2.1 Définitions

– Une ***application*** (ou une ***fonction***) $f : E \to F$, c'est la donnée pour chaque élément $x \in E$ d'un unique élément de F noté $f(x)$.
Nous représenterons les applications par deux types d'illustrations : les ensembles «patates», l'ensemble de départ (et celui d'arrivée) est schématisé par un ovale ses éléments par des points. L'association $x \mapsto f(x)$ est représentée par une flèche.

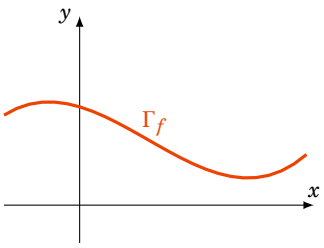


L'autre représentation est celle des fonctions continues de $\mathbb{R}$ dans $\mathbb{R}$ (ou des sous-ensembles de $\mathbb{R}$). L'ensemble de départ $\mathbb{R}$ est représenté par l'axe des abscisses et celui d'arrivée par l'axe des ordonnées. L'association $x \mapsto f(x)$ est représentée par le point $(x, f(x))$.

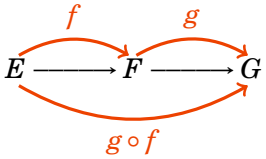


– ***Égalité***. Deux applications $f, g : E \to F$ sont égales si et seulement si pour tout $x \in E$, $f(x) = g(x)$. On note alors $f = g$.
– Le ***graphe*** de $f : E \to F$ est

$$\boxed{\Gamma_f = \Big\{\big(x, f(x)\big) \in E \times F \mid x \in E\Big\}}$$



– ***Composition***. Soient $f : E \to F$ et $g : F \to G$ alors $g \circ f : E \to G$ est l'application définie par $g \circ f(x) = g\big(f(x)\big)$.



Exemple 26.

1. L'***identité***, $\mathrm{id}_E : E \to E$ est simplement définie par $x \mapsto x$ et sera très utile dans la suite.
2. Définissons f, g ainsi

$$f : \begin{array}{ccc}]0,+\infty[& \longrightarrow &]0,+\infty[\\ x & \longmapsto & \frac{1}{x} \end{array}, \qquad g : \begin{array}{ccc}]0,+\infty[& \longrightarrow & \mathbb{R} \\ x & \longmapsto & \frac{x-1}{x+1} \end{array}.$$

Alors $g \circ f :]0,+\infty[\to \mathbb{R}$ vérifie pour tout $x \in]0,+\infty[$:

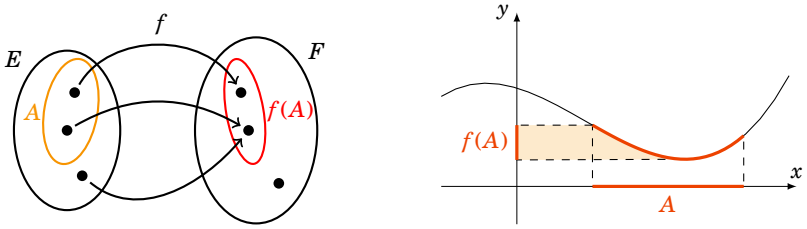
$$g \circ f(x) = g\big(f(x)\big) = g\left(\frac{1}{x}\right) = \frac{\frac{1}{x}-1}{\frac{1}{x}+1} = \frac{1-x}{1+x} = -g(x).$$

2.2 Image directe, image réciproque

Soient E, F deux ensembles.

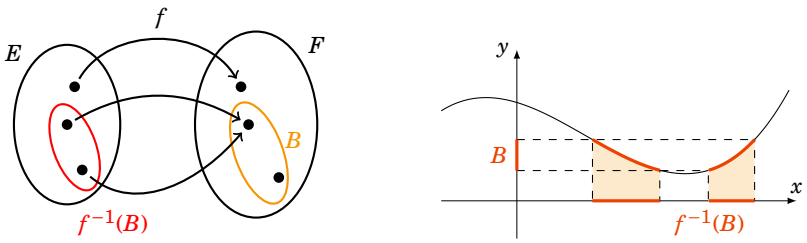
Définition 8. Soit $A \subset E$ et $f : E \to F$, l'***image directe*** de A par f est l'ensemble

$$f(A) = \{f(x) \mid x \in A\}$$



Définition 9. Soit $B \subset F$ et $f : E \to F$, l'***image réciproque*** de B par f est l'ensemble

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

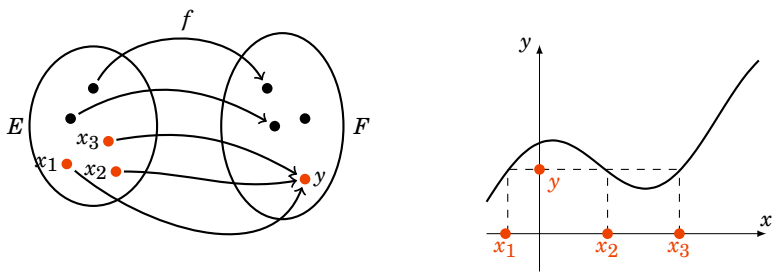


Remarque. Ces notions sont plus difficiles à maîtriser qu'il n'y paraît !
- $f(A)$ est un sous-ensemble de F, $f^{-1}(B)$ est un sous-ensemble de E.
- La notation «$f^{-1}(B)$» est un tout, rien ne dit que f est un fonction bijective (voir plus loin). L'image réciproque existe quelque soit la fonction.
- L'image directe d'un singleton $f(\{x\}) = \{f(x)\}$ est un singleton. Par contre l'image réciproque d'un singleton $f^{-1}(\{y\})$ dépend de f. Cela peut être un singleton, un ensemble à plusieurs éléments ; mais cela peut-être E tout entier (si f est une fonction constante) ou même l'ensemble vide (si aucune image par f ne vaut y).

2.3 Antécédents

Fixons $y \in F$. Tout élément $x \in E$ tel que $f(x) = y$ est un ***antécédent*** de y.
En termes d'image réciproque l'ensemble des antécédents de y est $f^{-1}(\{y\})$.

Sur les dessins suivants, l'élément y admet 3 antécédents par f. Ce sont x_1, x_2, x_3.



2.4 Mini-exercices

1. Pour deux applications $f, g : E \to F$, quelle est la négation de $f = g$?
2. Représenter le graphe de $f : \mathbb{N} \to \mathbb{R}$ définie par $n \mapsto \frac{4}{n+1}$.
3. Soient $f, g, h : \mathbb{R} \to \mathbb{R}$ définies par $f(x) = x^2$, $g(x) = 2x+1$, $h(x) = x^3 - 1$. Calculer $f \circ (g \circ h)$ et $(f \circ g) \circ h$.
4. Pour la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par $x \mapsto x^2$ représenter et calculer les ensembles suivants : $f([0,1[)$, $f(\mathbb{R})$, $f(]-1,2[)$, $f^{-1}([1,2[)$, $f^{-1}([-1,1])$, $f^{-1}(\{3\})$, $f^{-1}(\mathbb{R} \setminus \mathbb{N})$.

3 Injection, surjection, bijection

3.1 Injection, surjection

Soit E, F deux ensembles et $f : E \to F$ une application.

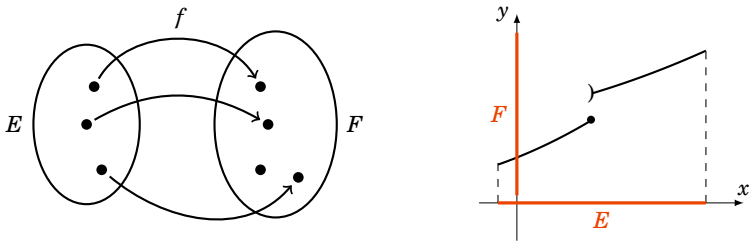
Définition 10. f est ***injective*** si pour tout $x, x' \in E$ avec $f(x) = f(x')$ alors $x = x'$. Autrement dit :

$$\forall x, x' \in E \quad \big(f(x) = f(x') \implies x = x'\big)$$

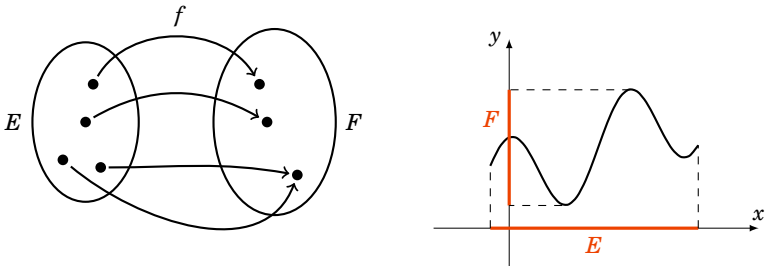
Définition 11. f est ***surjective*** si pour tout $y \in F$, il existe $x \in E$ tel que $y = f(x)$. Autrement dit :

$$\forall y \in F \quad \exists x \in E \quad \big(y = f(x)\big)$$

Une autre formulation : f est surjective si et seulement si $f(E) = F$.
Les applications f représentées sont injectives :



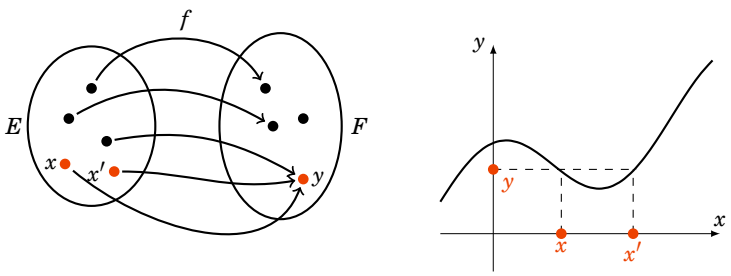
Les applications f représentées sont surjectives :



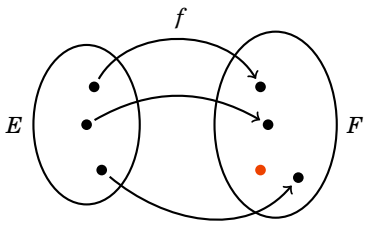
Remarque. Encore une fois ce sont des notions difficiles à appréhender. Une autre façon de formuler l'injectivité et la surjectivité est d'utiliser les antécédents.

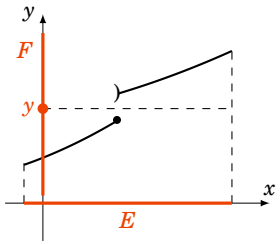
- f est injective si et seulement si tout élément y de F a *au plus* 1 antécédent (et éventuellement aucun).
- f est surjective si et seulement si tout élément y de F a *au moins* 1 antécédent.

Remarque. Voici deux fonctions non injectives :



Ainsi que deux fonctions non surjectives :





Exemple 27.

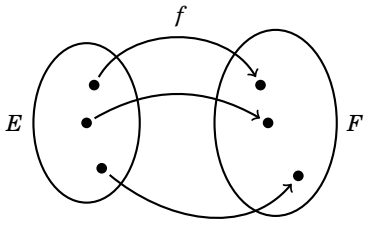
1. Soit $f_1 : \mathbb{N} \to \mathbb{Q}$ définie par $f_1(x) = \frac{1}{1+x}$. Montrons que f_1 est injective : soit $x, x' \in \mathbb{N}$ tels que $f_1(x) = f_1(x')$. Alors $\frac{1}{1+x} = \frac{1}{1+x'}$, donc $1+x = 1+x'$ et donc $x = x'$. Ainsi f_1 est injective.
   Par contre f_1 n'est pas surjective. Il s'agit de trouver un élément y qui n'a pas d'antécédent par f_1. Ici il est facile de voir que l'on a toujours $f_1(x) \leqslant 1$ et donc par exemple $y = 2$ n'a pas d'antécédent. Ainsi f_1 n'est pas surjective.
2. Soit $f_2 : \mathbb{Z} \to \mathbb{N}$ définie par $f_2(x) = x^2$. Alors f_2 n'est pas injective. En effet on peut trouver deux éléments $x, x' \in \mathbb{Z}$ différents tels que $f_2(x) = f_2(x')$. Il suffit de prendre par exemple $x = 2$, $x' = -2$.
   f_2 n'est pas non plus surjective, en effet il existe des éléments $y \in \mathbb{N}$ qui n'ont aucun antécédent. Par exemple $y = 3$: si $y = 3$ avait un antécédent x par f_2, nous aurions $f_2(x) = y$, c'est-à-dire $x^2 = 3$, d'où $x = \pm\sqrt{3}$. Mais alors x n'est pas un entier de $\mathbb{Z}$. Donc $y = 3$ n'a pas d'antécédent et f_2 n'est pas surjective.

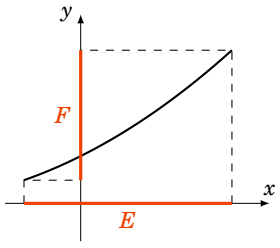
3.2 Bijection

Définition 12. f est ***bijective*** si elle injective et surjective. Cela équivaut à : pour tout $y \in F$ il existe un unique $x \in E$ tel que $y = f(x)$. Autrement dit :

$$\forall y \in F \quad \exists! x \in E \quad \big(y = f(x)\big)$$

L'existence du x vient de la surjectivité et l'unicité de l'injectivité. Autrement dit, tout élément de F a un unique antécédent par f.





Proposition 10.
Soit E, F des ensembles et $f : E \to F$ une application.

1. L'application f est bijective si et seulement si il existe une application $g : F \to E$ telle que $f \circ g = \mathrm{id}_F$ et $g \circ f = \mathrm{id}_E$.
2. Si f est bijective alors l'application g est unique et elle aussi est bijective. L'application g s'appelle la ***bijection réciproque*** de f et est notée f^{-1}. De plus $\big(f^{-1}\big)^{-1} = f$.

Remarque.

– $f \circ g = \mathrm{id}_F$ se reformule ainsi

$$\forall y \in F \quad f\big(g(y)\big) = y.$$

– Alors que $g \circ f = \mathrm{id}_E$ s'écrit :

$$\forall x \in E \quad g\big(f(x)\big) = x.$$

– Par exemple $f : \mathbb{R} \to]0,+\infty[$ définie par $f(x) = \exp(x)$ est bijective, sa bijection réciproque est $g :]0,+\infty[\to \mathbb{R}$ définie par $g(y) = \ln(y)$. Nous avons bien $\exp\big(\ln(y)\big) = y$, pour tout $y \in]0,+\infty[$ et $\ln\big(\exp(x)\big) = x$, pour tout $x \in \mathbb{R}$.

Démonstration.

1. – Sens $\Rightarrow$. Supposons f bijective. Nous allons construire une application $g : F \to E$. Comme f est surjective alors pour chaque $y \in F$, il existe un $x \in E$ tel que $y = f(x)$ et on pose $g(y) = x$. On a $f\big(g(y)\big) = f(x) = y$, ceci pour tout $y \in F$ et donc $f \circ g = \mathrm{id}_F$. On compose à droite avec f donc $f \circ g \circ f = \mathrm{id}_F \circ f$. Alors pour tout $x \in E$ on a $f\big(g \circ f(x)\big) = f(x)$ or f est injective et donc $g \circ f(x) = x$. Ainsi $g \circ f = \mathrm{id}_E$. Bilan : $f \circ g = \mathrm{id}_F$ et $g \circ f = \mathrm{id}_E$.
   – Sens $\Leftarrow$. Supposons que g existe et montrons que f est bijective.
     – f est surjective : en effet soit $y \in F$ alors on note $x = g(y) \in E$; on a bien : $f(x) = f\big(g(y)\big) = f \circ g(y) = \mathrm{id}_F(y) = y$, donc f est bien surjective.
     – f est injective : soient $x, x' \in E$ tels que $f(x) = f(x')$. On compose par g (à gauche) alors $g \circ f(x) = g \circ f(x')$ donc $\mathrm{id}_E(x) = \mathrm{id}_E(x')$ donc $x = x'$; f est bien injective.
2. – Si f est bijective alors g est aussi bijective car $g \circ f = \mathrm{id}_E$ et $f \circ g = \mathrm{id}_F$ et on applique ce que l'on vient de démontrer avec g à la place de f. Ainsi $g^{-1} = f$.
   – Si f est bijective, g est unique : en effet soit $h : F \to E$ une autre application telle que $h \circ f = \mathrm{id}_E$ et $f \circ h = \mathrm{id}_F$; en particulier $f \circ h = \mathrm{id}_F = f \circ g$, donc pour tout $y \in F$, $f\big(h(y)\big) = f\big(g(y)\big)$ or f est injective alors $h(y) = g(y)$, ceci pour tout $y \in F$; d'où $h = g$.

□

Proposition 11.
Soient $f : E \to F$ et $g : F \to G$ des applications bijectives. L'application $g \circ f$ est bijective et sa bijection réciproque est

$$\boxed{(g \circ f)^{-1} = f^{-1} \circ g^{-1}}$$

Démonstration. D'après la proposition 10, il existe $u : F \to E$ tel que $u \circ f = \mathrm{id}_E$ et $f \circ u = \mathrm{id}_F$. Il existe aussi $v : G \to F$ tel que $v \circ g = \mathrm{id}_F$ et $g \circ v = \mathrm{id}_G$. On a alors $(g \circ f) \circ (u \circ v) = g \circ (f \circ u) \circ v = g \circ \mathrm{id}_F \circ u = g \circ u = \mathrm{id}_E$. Et $(u \circ v) \circ (g \circ f) = u \circ (v \circ g) \circ f = u \circ \mathrm{id}_F \circ f = u \circ f = \mathrm{id}_E$. Donc $g \circ f$ est bijective et son inverse est $u \circ v$. Comme u est la bijection réciproque de f et v celle de g alors : $u \circ v = f^{-1} \circ g^{-1}$. □

3.3 Mini-exercices

1. Les fonctions suivantes sont-elles injectives, surjectives, bijectives ?
   – $f_1 : \mathbb{R} \to [0,+\infty[$, $x \mapsto x^2$.
   – $f_2 : [0,+\infty[\to [0,+\infty[$, $x \mapsto x^2$.
   – $f_3 : \mathbb{N} \to \mathbb{N}$, $x \mapsto x^2$.
   – $f_4 : \mathbb{Z} \to \mathbb{Z}$, $x \mapsto x - 7$.
   – $f_5 : \mathbb{R} \to [0,+\infty[$, $x \mapsto |x|$.
2. Montrer que la fonction $f :]1,+\infty[\to]0,+\infty[$ définie par $f(x) = \frac{1}{x-1}$ est bijective. Calculer sa bijection réciproque.

4 Ensembles finis

4.1 Cardinal

Définition 13. Un ensemble E est ***fini*** s'il existe un entier $n \in \mathbb{N}$ et une bijection de E vers $\{1,2,\ldots,n\}$. Cet entier n est unique et s'appelle le ***cardinal*** de E (ou le ***nombre d'éléments***) et est noté $\mathrm{Card}\, E$.

Quelques exemples :

1. $E = \{\text{rouge}, \text{noir}\}$ est en bijection avec $\{1,2\}$ et donc est de cardinal 2.

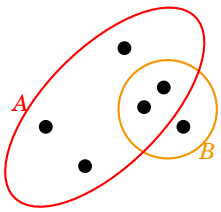
2. $\mathbb{N}$ n'est pas un ensemble fini.
3. Par définition le cardinal de l'ensemble vide est 0.

Enfin quelques propriétés :

1. Si A est un ensemble fini et $B \subset A$ alors B est un ensemble fini et $\operatorname{Card} B \leqslant \operatorname{Card} A$.
2. Si A, B sont des ensembles finis disjoints (c'est-à-dire $A \cap B = \varnothing$) alors $\operatorname{Card}(A \cup B) = \operatorname{Card} A + \operatorname{Card} B$.
3. Si A est un ensemble fini et $B \subset A$ alors $\operatorname{Card}(A \setminus B) = \operatorname{Card} A - \operatorname{Card} B$.
4. Enfin pour A, B deux ensembles finis quelconques :

$$\boxed{\operatorname{Card}(A \cup B) = \operatorname{Card} A + \operatorname{Card} B - \operatorname{Card}(A \cap B)}$$

Voici une situation où s'applique la dernière propriété :



4.2 Injection, surjection, bijection et ensembles finis

Proposition 12.
Soit E, F deux ensembles finis et $f : E \to F$ une application.

1. Si f est injective alors $\operatorname{Card} E \leqslant \operatorname{Card} F$.
2. Si f est surjective alors $\operatorname{Card} E \geqslant \operatorname{Card} F$.
3. Si f est bijective alors $\operatorname{Card} E = \operatorname{Card} F$.

Démonstration.

1. Supposons f injective. Notons $F' = f(E) \subset F$ alors la restriction $f_| : E \to F'$ (définie par $f_|(x) = f(x)$) est une bijection. Donc pour chaque $y \in F'$ est associé un unique $x \in E$ tel que $y = f(x)$. Donc E et F' ont le même nombre d'éléments. Donc $\operatorname{Card} F' = \operatorname{Card} E$. Or $F' \subset F$, ainsi $\operatorname{Card} E = \operatorname{Card} F' \leqslant \operatorname{Card} F$.
2. Supposons f surjective. Pour tout élément $y \in F$, il existe au moins un élément x de E tel que $y = f(x)$ et donc $\operatorname{Card} E \geqslant \operatorname{Card} F$.
3. Cela découle de (1) et (2) (ou aussi de la preuve du (1)).

□

Proposition 13.
Soit E, F deux ensembles finis et $f : E \to F$ une application. Si

$$\operatorname{Card} E = \operatorname{Card} F$$

alors les assertions suivantes sont équivalentes :

i. f est injective,
ii. f est surjective,
iii. f est bijective.

Démonstration. Le schéma de la preuve est le suivant : nous allons montrer successivement les implications :

$$(i) \Longrightarrow (ii) \Longrightarrow (iii) \Longrightarrow (i)$$

ce qui prouvera bien toutes les équivalences.

– $(i) \implies (ii)$. Supposons f injective. Alors $\operatorname{Card} f(E) = \operatorname{Card} E = \operatorname{Card} F$. Ainsi $f(E)$ est un sous-ensemble de F ayant le même cardinal que F ; cela entraîne $f(E) = F$ et donc f est surjective.
– $(ii) \implies (iii)$. Supposons f surjective. Pour montrer que f est bijective, il reste à montrer que f est injective. Raisonnons par l'absurde et supposons f non injective. Alors $\operatorname{Card} f(E) < \operatorname{Card} E$ (car au moins 2 éléments ont la même image). Or $f(E) = F$ car f surjective, donc $\operatorname{Card} F < \operatorname{Card} E$. C'est une contradiction, donc f doit être injective et ainsi f est bijective.
– $(iii) \implies (i)$. C'est clair : une fonction bijective est en particulier injective.

□

Appliquez ceci pour montrer le ***principe des tiroirs*** :

Proposition 14.
Si l'on range dans k tiroirs, $n > k$ paires de chaussettes alors il existe (au moins) un tiroir contenant (au moins) deux paires de chaussettes.

Malgré sa formulation amusante, c'est une proposition souvent utile. Exemple : dans un amphi de 400 étudiants, il y a au moins deux étudiants nés le même jour !

4.3 Nombres d'applications

Soient E, F des ensembles finis, non vides. On note $\operatorname{Card} E = n$ et $\operatorname{Card} F = p$.

Proposition 15.
Le nombre d'applications différentes de E dans F est :

$$\boxed{p^n}$$

Autrement dit c'est $\boxed{(\operatorname{Card} F)^{\operatorname{Card} E}}$.

Exemple 28. En particulier le nombre d'applications de E dans lui-même est n^n. Par exemple si $E = \{1,2,3,4,5\}$ alors ce nombre est $5^5 = 3125$.

Démonstration. Fixons F et $p = \operatorname{Card} F$. Nous allons effectuer une récurrence sur $n = \operatorname{Card} E$. Soit (P_n) l'assertion suivante : le nombre d'applications d'un ensemble à n éléments vers un ensemble à p éléments est p^n.
– *Initialisation.* Pour $n = 1$, une application de E dans F est définie par l'image de l'unique élément de E. Il y a $p = \operatorname{Card} F$ choix possibles et donc p^1 applications distinctes. Ainsi P_1 est vraie.
– *Hérédité.* Fixons $n \geqslant 1$ et supposons que P_n est vraie. Soit E un ensemble à $n+1$ éléments. On choisit et fixe $a \in E$; soit alors $E' = E \setminus \{a\}$ qui a bien n éléments. Le nombre d'applications de E' vers F est p^n, par l'hypothèse de récurrence (P_n). Pour chaque application $f : E' \to F$ on peut la prolonger en une application $f : E \to F$ en choisissant l'image de a. On a p choix pour l'image de a et donc $p^n \times p$ choix pour les applications de E vers F. Ainsi P_{n+1} est vérifiée.
– *Conclusion.* Par le principe de récurrence P_n est vraie, pour tout $n \geqslant 1$.

□

Proposition 16.
Le nombre d'injections de E dans F est :

$$p \times (p-1) \times \cdots \times (p-(n-1)).$$

Démonstration. Supposons $E = \{a_1, a_2, \ldots, a_n\}$; pour l'image de a_1 nous avons p choix. Une fois ce choix fait, pour l'image de a_2 il reste $p-1$ choix (car a_2 ne doit pas avoir la même image que a_1). Pour l'image de a_3 il y a $p-2$ possibilités. Ainsi de suite : pour l'image de a_k il y $p-(k-1)$ choix... Il y a au final $p \times (p-1) \times \cdots \times (p-(n-1))$ applications injectives. □

Notation ***factorielle*** : $n! = 1 \times 2 \times 3 \times \cdots \times n$. Avec $1! = 1$ et par convention $0! = 1$.

Proposition 17.
Le nombre de bijections d'un ensemble E de cardinal n dans lui-même est :

$$\boxed{n!}$$

Exemple 29. Parmi les 3125 applications de $\{1,2,3,4,5\}$ dans lui-même il y en a $5! = 120$ qui sont bijectives.

Démonstration. Nous allons le prouver par récurrence sur n. Soit (P_n) l'assertion suivante : le nombre de bijections d'un ensemble à n éléments dans un ensemble à n éléments est $n!$
- P_1 est vraie. Il n'y a qu'une bijection d'un ensemble à 1 élément dans un ensemble à 1 élément.
- Fixons $n \geqslant 1$ et supposons que P_n est vraie. Soit E un ensemble à $n+1$ éléments. On fixe $a \in E$. Pour chaque $b \in E$ il y a -par l'hypothèse de récurrence- exactement $n!$ applications bijectives de $E \setminus \{a\} \to E \setminus \{b\}$. Chaque application se prolonge en une bijection de $E \to F$ en posant $a \mapsto b$. Comme il y a $n+1$ choix de $b \in E$ alors nous obtenons $n! \times (n+1)$ bijections de E dans lui-même. Ainsi P_{n+1} est vraie.
- Par le principe de récurrence le nombre de bijections d'un ensemble à n éléments est $n!$

On aurait aussi pu directement utiliser la proposition 16 avec $n = p$ (sachant qu'alors les injections sont aussi des bijections). □

4.4 Nombres de sous-ensembles

Soit E un ensemble fini de cardinal n.

Proposition 18.
Il y a $2^{\operatorname{Card} E}$ sous-ensembles de E :

$$\boxed{\operatorname{Card} \mathscr{P}(E) = 2^n}$$

Exemple 30. Si $E = \{1,2,3,4,5\}$ alors $\mathscr{P}(E)$ a $2^5 = 32$ parties. C'est un bon exercice de les énumérer :
- l'ensemble vide : $\varnothing$,
- 5 singletons : $\{1\},\{2\},\ldots$,
- 10 paires : $\{1,2\},\{1,3\},\ldots,\{2,3\},\ldots$,
- 10 triplets : $\{1,2,3\},\ldots$,
- 5 ensembles à 4 éléments : $\{1,2,3,4\},\{1,2,3,5\},\ldots$,
- et E tout entier : $\{1,2,3,4,5\}$.

Démonstration. Encore une récurrence sur $n = \operatorname{Card} E$.
- Si $n = 1$, $E = \{a\}$ est un singleton, les deux sous-ensembles sont : $\varnothing$ et E.
- Supposons que la proposition soit vraie pour $n \geqslant 1$ fixé. Soit E un ensemble à $n+1$ éléments. On fixe $a \in E$. Il y a deux sortes de sous-ensembles de E :
  - les sous-ensembles A qui ne contiennent pas a : ce sont les sous-ensembles $A \subset E \setminus \{a\}$. Par l'hypothèse de récurrence il y en a 2^n.
  - les sous-ensembles A qui contiennent a : ils sont de la forme $A = \{a\} \cup A'$ avec $A' \subset E \setminus \{a\}$. Par l'hypothèse de récurrence il y a 2^n sous-ensembles A' possibles et donc aussi 2^n sous-ensembles A.

  Le bilan : $2^n + 2^n = 2^{n+1}$ parties $A \subset E$.
- Par le principe de récurrence, nous avons prouvé que si $\operatorname{Card} E = n$ alors $\operatorname{Card} \mathscr{P}(E) = 2^n$.

□

4.5 Coefficients du binôme de Newton

Définition 14. Le nombre de parties à k éléments d'un ensemble à n éléments est noté $\binom{n}{k}$ ou C_n^k.

Exemple 31. Les parties à deux éléments de $\{1,2,3\}$ sont $\{1,2\}$, $\{1,3\}$ et $\{2,3\}$ et donc $\binom{3}{2} = 3$. Nous avons déjà classé les parties de $\{1,2,3,4,5\}$ par nombre d'éléments et donc
- $\binom{5}{0} = 1$ (la seule partie n'ayant aucun élément est l'ensemble vide),

– $\binom{5}{1} = 5$ (il y a 5 singletons),
– $\binom{5}{2} = 10$ (il y a 10 paires),
– $\binom{5}{3} = 10$,
– $\binom{5}{4} = 5$,
– $\binom{5}{5} = 1$ (la seule partie ayant 5 éléments est l'ensemble tout entier).

Sans calculs on peut déjà remarquer les faits suivants :

Proposition 19.

– $\binom{n}{0} = 1$, $\binom{n}{1} = n$, $\binom{n}{n} = 1$.

– $\boxed{\binom{n}{n-k} = \binom{n}{k}}$

– $\boxed{\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{k} + \cdots + \binom{n}{n} = 2^n}$

Démonstration.

1. Par exemple : $\binom{n}{1} = n$ car il y a n singletons.
2. Compter le nombre de parties $A \subset E$ ayant k éléments revient aussi à compter le nombre de parties de la forme $\complement A$ (qui ont donc $n-k$ éléments), ainsi $\binom{n}{n-k} = \binom{n}{k}$.
3. La formule $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{k} + \cdots + \binom{n}{n} = 2^n$ exprime que faire la somme du nombre de parties à k éléments, pour $k = 0, \ldots, n$, revient à compter toutes les parties de E.

□

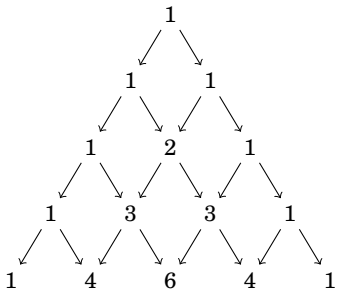
Proposition 20.

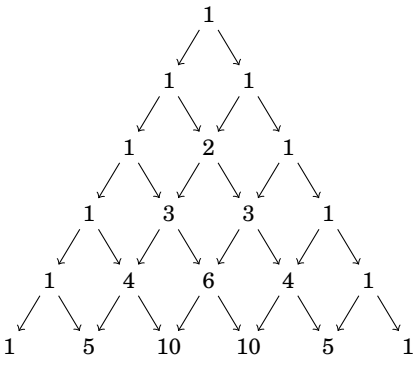
$$\boxed{\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \quad 0 < k < n}$$

Démonstration. Soit E un ensemble à n éléments, $a \in E$ et $E' = E \setminus \{a\}$. Il y a deux sortes de parties $A \subset E$ ayant k éléments :
– celles qui ne contiennent pas a : ce sont donc des parties à k éléments dans E' qui a $n-1$ éléments. Il y a en a donc $\binom{n-1}{k}$,
– celles qui contiennent a : elles sont de la forme $A = \{a\} \cup A'$ avec A' une partie à $k-1$ éléments dans E' qui a $n-1$ éléments. Il y en a $\binom{n-1}{k-1}$.

Bilan : $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. □

Le triangle de Pascal est un algorithme pour calculer ces coefficients $\binom{n}{k}$. La ligne du haut correspond à $\binom{0}{0}$, la ligne suivante à $\binom{1}{0}$ et $\binom{1}{1}$, la ligne d'après à $\binom{2}{0}$, $\binom{2}{1}$ et $\binom{2}{2}$.
La dernière ligne du triangle de gauche aux coefficients $\binom{4}{0}$, $\binom{4}{1}$, …, $\binom{4}{4}$.
Comment continuer ce triangle pour obtenir le triangle de droite ? Chaque élément de la nouvelle ligne est obtenu en ajoutant les deux nombres qui lui sont au-dessus à droite et au-dessus à gauche.





Ce qui fait que cela fonctionne c'est bien sûr la proposition 20 qui se représente ainsi :

$$\begin{array}{ccc} \binom{n-1}{k-1} & & \binom{n-1}{k} \\ & \searrow \quad \swarrow & \\ & \binom{n}{k} & \end{array}$$

Une autre façon de calculer le coefficient du binôme de Newton repose sur la formule suivante :

Proposition 21.

$$\boxed{\binom{n}{k} = \frac{n!}{k!(n-k)!}}$$

Démonstration. Cela se fait par récurrence sur n. C'est clair pour $n = 1$. Si c'est vrai au rang $n-1$ alors écrivons $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ et utilisons l'hypothèse de récurrence pour $\binom{n-1}{k-1}$ et $\binom{n-1}{k}$. Ainsi

$$\begin{aligned} \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} &= \frac{(n-1)!}{(k-1)!(n-1-(k-1))!} + \frac{(n-1)!}{k!(n-1-k)!} \\ &= \frac{(n-1)!}{(k-1)!(n-k-1)!} \times \left(\frac{1}{n-k} + \frac{1}{k}\right) = \frac{(n-1)!}{(k-1)!(n-k-1)!} \times \frac{n}{k(n-k)} \\ &= \frac{n!}{k!(n-k)!} \end{aligned}$$

□

4.6 Formule du binôme de Newton

Théorème 7.
Soient $a, b \in \mathbb{R}$ et n un entier positif alors :

$$\boxed{(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} \cdot b^k}$$

Autrement dit :

$$(a+b)^n = \binom{n}{0} a^n \cdot b^0 + \binom{n}{1} a^{n-1} \cdot b^1 + \cdots + \binom{n}{k} a^{n-k} \cdot b^k + \cdots + \binom{n}{n} a^0 \cdot b^n$$

Le théorème est aussi vrai si a et b sont des nombres complexes.

Exemple 32.

1. Pour $n = 2$ on retrouve la formule archi-connue : $(a+b)^2 = a^2 + 2ab + b^2$.
2. Il est aussi bon de connaître $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$.
3. Si $a = 1$ et $b = 1$ on retrouve la formule : $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.

Démonstration. Nous allons effectuer une récurrence sur n. Soit (P_n) l'assertion : $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} \cdot b^k$.
– *Initialisation.* Pour $n = 1$, $(a+b)^1 = \binom{1}{0} a^1 b^0 + \binom{1}{1} a^0 b^1$. Ainsi P_1 est vraie.

– *Hérédité.* Fixons $n \geqslant 2$ et supposons que P_{n-1} est vraie.

$$\begin{aligned}(a+b)^n = (a+b)\cdot(a+b)^{n-1} &= a\left(a^{n-1}+\cdots+\binom{n-1}{k}a^{n-1-k}b^k+\cdots+b^{n-1}\right)\\ &\quad +b\left(a^{n-1}+\cdots+\binom{n-1}{k-1}a^{n-1-(k-1)}b^{k-1}+\cdots+b^{n-1}\right)\\ &= \cdots+\left(\binom{n-1}{k}+\binom{n-1}{k-1}\right)a^{n-k}b^k+\cdots\\ &= \cdots+\binom{n}{k}a^{n-k}b^k+\cdots=\sum_{k=0}^{n}\binom{n}{k}a^{n-k}\cdot b^k\end{aligned}$$

Ainsi P_{n+1} est vérifiée.

– *Conclusion.* Par le principe de récurrence P_n est vraie, pour tout $n \geqslant 1$.

□

4.7 Mini-exercices

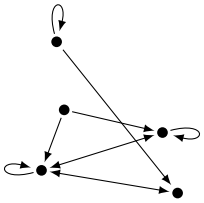
1. Combien y a-t-il d'applications injectives d'un ensemble à n éléments dans un ensemble à $n+1$ éléments ?
2. Combien y a-t-il d'applications surjectives d'un ensemble à $n+1$ éléments dans un ensemble à n éléments ?
3. Calculer le nombre de façons de choisir 5 cartes dans un jeux de 32 cartes.
4. Calculer le nombre de listes à k éléments dans un ensemble à n éléments (les listes sont ordonnées : par exemple $(1,2,3) \neq (1,3,2)$).
5. Développer $(a-b)^4$, $(a+b)^5$.
6. Que donne la formule du binôme pour $a=-1$, $b=+1$? En déduire que dans un ensemble à n éléments il y a autant de parties de cardinal pair que de cardinal impair.

5 Relation d'équivalence

5.1 Définition

Une ***relation*** sur un ensemble E, c'est la donnée pour tout couple $(x,y) \in E \times E$ de «Vrai» (s'ils sont en relation), ou de «Faux» sinon.

Nous schématisons une relation ainsi : les éléments de E sont des points, une flèche de x vers y signifie que x est en relation avec y, c'est-à-dire que l'on associe «Vrai» au couple (x,y).

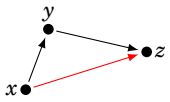


Définition 15. Soit E un ensemble et $\mathcal{R}$ une relation, c'est une ***relation d'équivalence*** si :

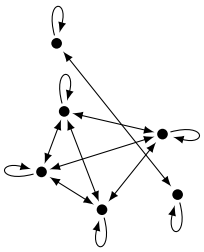
– $\forall x \in E,\ x\mathcal{R}x$, (***réflexivité***)

– $\forall x,y \in E,\ x\mathcal{R}y \implies y\mathcal{R}x$, (***symétrie***)

– $\forall x, y, z \in E$, $x \mathscr{R} y$ et $y \mathscr{R} z \implies x \mathscr{R} z$, (***transitivité***)



Exemple de relation d'équivalence :



5.2 Exemples

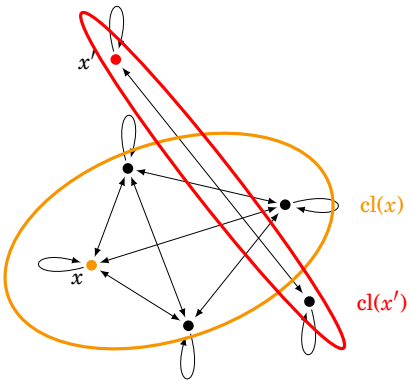
Exemple 33. Voici des exemples basiques.

1. La relation $\mathscr{R}$ «être parallèle» est une relation d'équivalence pour l'ensemble E des droites affines du plan.
   – réflexivité : une droite est parallèle à elle-même,
   – symétrie : si D est parallèle à D' alors D' est parallèle à D,
   – transitivité : si D parallèle à D' et D' parallèle à D'' alors D est parallèle à D''.
2. La relation «être du même âge» est une relation d'équivalence.
3. La relation «être perpendiculaire» n'est pas une relation d'équivalence (ni la réflexivité, ni la transitivité ne sont vérifiées).
4. La relation $\leqslant$ (sur $E = \mathbb{R}$ par exemple) n'est pas une relation d'équivalence (la symétrie n'est pas vérifiée).

5.3 Classes d'équivalence

Définition 16. Soit $\mathscr{R}$ une relation d'équivalence sur un ensemble E. Soit $x \in E$, la ***classe d'équivalence*** de x est

$$\boxed{\mathrm{cl}(x) = \{y \in E \mid y \mathscr{R} x\}}$$



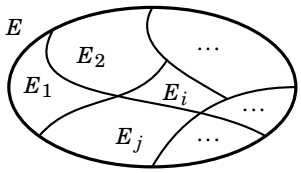
$\mathrm{cl}(x)$ est donc un sous-ensemble de E, on le note aussi $\overline{x}$. Si $y \in \mathrm{cl}(x)$, on dit que y un ***représentant*** de $\mathrm{cl}(x)$.

Soit E un ensemble et $\mathscr{R}$ une relation d'équivalence.

Proposition 22.
On a les propriétés suivantes :

1. $\mathrm{cl}(x) = \mathrm{cl}(y) \iff x \mathscr{R} y$.
2. Pour tout $x, y \in E$, $\mathrm{cl}(x) = \mathrm{cl}(y)$ ou $\mathrm{cl}(x) \cap \mathrm{cl}(y) = \varnothing$.
3. Soit C un ensemble de représentants de toutes les classes alors $\{\mathrm{cl}(x) \mid x \in C\}$ constitue une partition de E.

Une ***partition*** de E est un ensemble $\{E_i\}$ de parties de E tel que $E = \bigcup_i E_i$ et $E_i \cap E_j = \varnothing$ (si $i \neq j$).



Exemples :

1. Pour la relation «être du même âge», la classe d'équivalence d'une personne est l'ensemble des personnes ayant le même âge. Il y a donc une classe d'équivalence formée des personnes de 19 ans, une autre formée des personnes de 20 ans,... Les trois assertions de la proposition se lisent ainsi :
   - On est dans la même classe d'équivalence si et seulement si on est du même âge.
   - Deux personnes appartiennent soit à la même classe, soit à des classes disjointes.
   - Si on choisit une personne de chaque âge possible, cela forme un ensemble de représentants C. Maintenant une personne quelconque appartient à une et une seule classe d'un des représentants.
2. Pour la relation «être parallèle», la classe d'équivalence d'une droite est l'ensemble des droites parallèles. À chaque classe d'équivalence correspond une et une seule direction.

Voici un exemple que vous connaissez depuis longtemps :

Exemple 34. Définissons sur $E = \mathbb{Z} \times \mathbb{N}^*$ la relation $\mathscr{R}$ par

$$(p,q) \mathscr{R} (p',q') \iff pq' = p'q.$$

Tout d'abord $\mathscr{R}$ est une relation d'équivalence :
- $\mathscr{R}$ est réflexive : pour tout (p,q) on a bien $pq = pq$ et donc $(p,q)\mathscr{R}(p,q)$.
- $\mathscr{R}$ est symétrique : pour tout (p,q), (p',q') tels que $(p,q)\mathscr{R}(p',q')$ on a donc $pq' = p'q$ et donc $p'q = pq'$ d'où $(p',q')\mathscr{R}(p,q)$.
- $\mathscr{R}$ est transitive : pour tout (p,q), (p',q'), (p'',q'') tels que $(p,q)\mathscr{R}(p',q')$ et $(p',q')\mathscr{R}(p'',q'')$ on a donc $pq' = p'q$ et $p'q'' = p''q'$. Alors $(pq')q'' = (p'q)q'' = q(p'q'') = q(p''q')$. En divisant par $q' \neq 0$ on obtient $pq'' = qp''$ et donc $(p,q)\mathscr{R}(p'',q'')$.

Nous allons noter $\frac{p}{q} = \mathrm{cl}(p,q)$ la classe d'équivalence d'un élément $(p,q) \in \mathbb{Z} \times \mathbb{N}^*$. Par exemple, comme $(2,3)\mathscr{R}(4,6)$ (car $2 \times 6 = 3 \times 4$) alors les classes de $(2,3)$ et $(4,6)$ sont égales : avec notre notation cela s'écrit : $\frac{2}{3} = \frac{4}{6}$.
C'est ainsi que l'on définit les rationnels : l'ensemble $\mathbb{Q}$ des rationnels est l'ensemble de classes d'équivalence de la relation $\mathscr{R}$.
Les nombres $\frac{2}{3} = \frac{4}{6}$ sont bien égaux (ce sont les mêmes classes) mais les écritures sont différentes (les représentants sont distincts).

5.4 L'ensemble $\mathbb{Z}/n\mathbb{Z}$

Soit $n \geqslant 2$ un entier. Définissons la relation suivante sur l'ensemble $E = \mathbb{Z}$:

$$a \equiv b \pmod{n} \quad \iff \quad a - b \text{ est un multiple de } n$$

Exemples pour $n = 7$: $10 \equiv 3 \pmod 7$, $19 \equiv 5 \pmod 7$, $77 \equiv 0 \pmod 7$, $-1 \equiv 20 \pmod 7$.

Cette relation est bien une relation d'équivalence :

– Pour tout $a \in \mathbb{Z}$, $a - a = 0 = 0 \cdot n$ est un multiple de n donc $a \equiv a \pmod{n}$.
– Pour $a, b \in \mathbb{Z}$ tels que $a \equiv b \pmod{n}$ alors $a - b$ est un multiple de n, autrement dit il existe $k \in \mathbb{Z}$ tel que $a - b = kn$ et donc $b - a = (-k)n$ et ainsi $b \equiv a \pmod{n}$.
– Si $a \equiv b \pmod{n}$ et $b \equiv c \pmod{n}$ alors il existe $k, k' \in \mathbb{Z}$ tels que $a - b = kn$ et $b - c = k'n$. Alors $a - c = (a - b) + (b - c) = (k + k')n$ et donc $a \equiv c \pmod{n}$.

La classe d'équivalence de $a \in \mathbb{Z}$ est notée $\overline{a}$. Par définition nous avons donc

$$\overline{a} = \mathrm{cl}(a) = \{b \in \mathbb{Z} \mid b \equiv a \pmod{n}\}.$$

Comme un tel b s'écrit $b = a + kn$ pour un certain $k \in \mathbb{Z}$ alors c'est aussi exactement

$$\overline{a} = a + n\mathbb{Z} = \{a + kn \mid k \in \mathbb{Z}\}.$$

Comme $n \equiv 0 \pmod{n}$, $n + 1 \equiv 1 \pmod{n}$, ... alors

$$\overline{n} = \overline{0}, \quad \overline{n+1} = \overline{1}, \quad \overline{n+2} = \overline{2}, \ldots$$

et donc l'ensemble des classes d'équivalence est l'ensemble

$$\boxed{\mathbb{Z}/n\mathbb{Z} = \{\overline{0}, \overline{1}, \overline{2}, \ldots, \overline{n-1}\}}$$

qui contient exactement n éléments.
Par exemple : pour $n = 7$, $\overline{0} = \{\ldots, -14, -7, 0, 7, 14, 21, \ldots\} = 7\mathbb{Z}$; $\overline{1} = \{\ldots, -13, -6, 1, 8, 15, \ldots\} = 1 + 7\mathbb{Z}$; ... ; $\overline{6} = \{\ldots, -8, -1, 6, 13, 20, \ldots\} = 6 + 7\mathbb{Z}$. Mais ensuite $\overline{7} = \{\ldots - 7, 0, 7, 14, 21, \ldots\} = \overline{0} = 7\mathbb{Z}$. Ainsi $\mathbb{Z}/7\mathbb{Z} = \{\overline{0}, \overline{1}, \overline{2}, \ldots, \overline{6}\}$ possède 7 éléments.

Remarque. Dans beaucoup de situations de la vie courante, nous raisonnons avec les modulos. Par exemple pour l'heure : les minutes et les secondes sont modulo 60 (après 59 minutes on repart à zéro), les heures modulo 24 (ou modulo 12 sur le cadran à aiguilles). Les jours de la semaine sont modulo 7, les mois modulo 12,...

5.5 Mini-exercices

1. Montrer que la relation définie sur $\mathbb{N}$ par $x \mathscr{R} y \iff \frac{2x+y}{3} \in \mathbb{N}$ est une relation d'équivalence. Montrer qu'il y a 3 classes d'équivalence.
2. Dans $\mathbb{R}^2$ montrer que la relation définie par $(x, y) \mathscr{R} (x', y') \iff x + y' = x' + y$ est une relation d'équivalence. Montrer que deux points (x, y) et (x', y') sont dans une même classe si et seulement s'ils appartiennent à une même droite dont vous déterminerez la direction.
3. On définit une addition sur $\mathbb{Z}/n\mathbb{Z}$ par $\overline{p} + \overline{q} = \overline{p+q}$. Calculer la table d'addition dans $\mathbb{Z}/6\mathbb{Z}$ (c'est-à-dire toutes les sommes $\overline{p} + \overline{q}$ pour $\overline{p}, \overline{q} \in \mathbb{Z}/6\mathbb{Z}$). Même chose avec la multiplication $\overline{p} \times \overline{q} = \overline{p \times q}$. Mêmes questions avec $\mathbb{Z}/5\mathbb{Z}$, puis $\mathbb{Z}/8\mathbb{Z}$.

Auteurs
Arnaud Bodin
Benjamin Boutin
Pascal Romon

Troisième partie

Les exercices

Arithmétique dans $\mathbb{Z}$

1 Divisibilité, division euclidienne

Exercice 1

Sachant que l'on a $96842 = 256 \times 375 + 842$, déterminer, sans faire la division, le reste de la division du nombre 96842 par chacun des nombres 256 et 375.

Indication ▼ Correction ▼ Vidéo ■ [000251]

Exercice 2

Montrer que $\forall n \in \mathbb{N}$:

$$n(n+1)(n+2)(n+3) \text{ est divisible par } 24,$$

$$n(n+1)(n+2)(n+3)(n+4) \text{ est divisible par } 120.$$

Correction ▼ Vidéo ■ [000257]

Exercice 3

Montrer que si n est un entier naturel somme de deux carrés d'entiers alors le reste de la division euclidienne de n par 4 n'est jamais égal à 3.

Correction ▼ Vidéo ■ [000267]

Exercice 4

Démontrer que le nombre $7^n + 1$ est divisible par 8 si n est impair ; dans le cas n pair, donner le reste de sa division par 8.

Indication ▼ Correction ▼ Vidéo ■ [000254]

Exercice 5

Trouver le reste de la division par 13 du nombre 100^{1000}.

Indication ▼ Correction ▼ Vidéo ■ [000250]

Exercice 6

1. Montrer que le reste de la division euclidienne par 8 du carré de tout nombre impair est 1.
2. Montrer de même que tout nombre pair vérifie $x^2 = 0 \pmod 8$ ou $x^2 = 4 \pmod 8$.
3. Soient a, b, c trois entiers impairs. Déterminer le reste modulo 8 de $a^2+b^2+c^2$ et celui de $2(ab+bc+ca)$.
4. En déduire que ces deux nombres ne sont pas des carrés puis que $ab+bc+ca$ non plus.

Indication ▼ Correction ▼ Vidéo ■ [000285]

2 pgcd, ppcm, algorithme d'Euclide

Exercice 7

Calculer le pgcd des nombres suivants :

1. 126, 230.
2. 390, 720, 450.
3. 180, 606, 750.

Correction ▼ Vidéo ■ [000290]

Exercice 8

Déterminer les couples d'entiers naturels de pgcd 18 et de somme 360. De même avec pgcd 18 et produit 6480.

Correction ▼ Vidéo ■ [000292]

Exercice 9

Calculer par l'algorithme d'Euclide : $\text{pgcd}(18480, 9828)$. En déduire une écriture de 84 comme combinaison linéaire de 18480 et 9828.

Correction ▼ Vidéo ■ [000296]

Exercice 10

Notons $a = 1\,111\,111\,111$ et $b = 123\,456\,789$.

1. Calculer le quotient et le reste de la division euclidienne de a par b.
2. Calculer $p = \text{pgcd}(a,b)$.
3. Déterminer deux entiers relatifs u et v tels que $au + bv = p$.

Correction ▼ Vidéo ■ [000303]

Exercice 11

Résoudre dans $\mathbb{Z}$: $1665x + 1035y = 45$.

Indication ▼ Correction ▼ Vidéo ■ [000305]

3 Nombres premiers, nombres premiers entre eux

Exercice 12

Combien 15! admet-il de diviseurs ?

Indication ▼ Correction ▼ Vidéo ■ [000249]

Exercice 13

Démontrer que, si a et b sont des entiers premiers entre eux, il en est de même des entiers $a+b$ et ab.

Indication ▼ Correction ▼ Vidéo ■ [000337]

Exercice 14

Soient a, b des entiers supérieurs ou égaux à 1. Montrer :

1. $(2^a - 1) | (2^{ab} - 1)$;
2. $2^p - 1$ premier $\Rightarrow$ p premier ;
3. $\text{pgcd}(2^a - 1, 2^b - 1) = 2^{\text{pgcd}(a,b)} - 1$.

Indication ▼ Correction ▼ Vidéo ■ [000336]

Exercice 15

Soit $a \in \mathbb{N}$ tel que $a^n + 1$ soit premier, montrer que $\exists k \in \mathbb{N}, n = 2^k$. Que penser de la conjecture : $\forall n \in \mathbb{N}, 2^{2^n} + 1$ est premier ?

Indication ▼ Correction ▼ Vidéo ■ [000349]

Exercice 16

Soit p un nombre premier.

1. Montrer que $\forall i \in \mathbb{N}, 0 < i < p$ on a :

$$C_p^i \text{ est divisible par } p.$$

2. Montrer par récurence que :

$$\forall p \text{ premier}, \forall a \in \mathbb{N}^*, \text{ on a } a^p - a \text{ est divisible par } p.$$

Indication ▼ Correction ▼ Vidéo ■ [000339]

Exercice 17

1. Montrer par récurrence que $\forall n \in \mathbb{N}, \forall k \geqslant 1$ on a :

$$2^{2^{n+k}} - 1 = \left(2^{2^n} - 1\right) \times \prod_{i=0}^{k-1} (2^{2^{n+i}} + 1).$$

2. On pose $F_n = 2^{2^n} + 1$. Montrer que pour $m \neq n$, F_n et F_m sont premiers entre eux.
3. En déduire qu'il y a une infinité de nombres premiers.

Indication ▼ Correction ▼ Vidéo ■ [000341]

Exercice 18

Soit X l'ensemble des nombres premiers de la forme $4k+3$ avec $k \in \mathbb{N}$.

1. Montrer que X est non vide.
2. Montrer que le produit de nombres de la forme $4k+1$ est encore de cette forme.
3. On suppose que X est fini et on l'écrit alors $X = \{p_1, \ldots, p_n\}$.
   Soit $a = 4p_1p_2 \ldots p_n - 1$. Montrer par l'absurde que a admet un diviseur premier de la forme $4k+3$.
4. Montrer que ceci est impossible et donc que X est infini.

Correction ▼ Vidéo ■ [000348]

Indication pour l'exercice 1 ▲

Attention le reste d'une division euclidienne est plus petit que le quotient !

Indication pour l'exercice 4 ▲

Utiliser les modulos (ici modulo 8), un entier est divisible par 8 si et seulement si il est équivalent à 0 modulo 8. Ici vous pouvez commencer par calculer $7^n \pmod 8$.

Indication pour l'exercice 5 ▲

Il faut travailler modulo 13, tout d'abord réduire 100 modulo 13. Se souvenir que si $a \equiv b \pmod{13}$ alors $a^k \equiv b^k \pmod{13}$. Enfin calculer ce que cela donne pour les exposants $k = 1, 2, 3, \ldots$ en essayant de trouver une règle générale.

Indication pour l'exercice 6 ▲

1. Écrire $n = 2p+1$.
2. Écrire $n = 2p$ et discuter selon que p est pair ou impair.
3. Utiliser la première question.
4. Par l'absurde supposer que cela s'écrive comme un carré, par exemple $a^2+b^2+c^2 = n^2$ puis discuter selon que n est pair ou impair.

Indication pour l'exercice 11 ▲

Commencer par simplifier l'équation ! Ensuite trouver une solution particulière (x_0, y_0) à l'aide de l'algorithme d'Euclide par exemple. Ensuite trouver un expression pour une solution générale.

Indication pour l'exercice 12 ▲

Il ne faut surtout pas chercher à calculer $15! = 1 \times 2 \times 3 \times 4 \times \cdots \times 15$, mais profiter du fait qu'il est déjà "presque" factorisé.

Indication pour l'exercice 13 ▲

Raisonner par l'absurde et utiliser le lemme de Gauss.

Indication pour l'exercice 14 ▲

Pour 1. utiliser l'égalité

$$x^b - 1 = (x-1)(x^{b-1} + \cdots + x + 1).$$

Pour 2. raisonner par contraposition et utiliser la question 1.
La question 3. est difficile ! Supposer $a \geq b$. Commencer par montrer que $\text{pgcd}(2^a-1, 2^b-1) = \text{pgcd}(2^a - 2^b, 2^b-1) = \text{pgcd}(2^{a-b}-1, 2^b-1)$. Cela vour permettra de comparer l'agorithme d'Euclide pour le calcul de $\text{pgcd}(a,b)$ avec l'algorithme d'Euclide pour le calcul de $\text{pgcd}(2^a-1, 2^b-1)$.

Indication pour l'exercice 15 ▲

Raisonner par contraposition (ou par l'absurde) : supposer que n n'est pas de la forme 2^k, alors n admet un facteur irréductible $p > 2$. Utiliser aussi $x^p + 1 = (x+1)(1 - x + x^2 - x^3 + \ldots + x^{p-1})$ avec x bien choisi.

Indication pour l'exercice 16 ▲

1. Écrire

$$C_p^i = \frac{p(p-1)(p-2)\ldots(p-(i+1))}{i!}$$

et utiliser le lemme de Gauss ou le lemme d'Euclide.

2. Raisonner avec les modulos, c'est-à-dire prouver $a^p \equiv a \pmod{p}$.

Indication pour l'exercice 17 ▲

1. Il faut être très soigneux : n est fixé une fois pour toute, la récurrence se fait sur $k \in \mathbb{N}$.
2. Utiliser la question précédente avec $m = n + k$.
3. Par l'absurde, supposer qu'il y a seulement N nombres premiers, considérer $N+1$ nombres du type F_i. Appliquer le "principe du tiroir" : *si vous avez $N+1$ chaussettes rangées dans N tiroirs alors il existe (au moins) un tiroir contenant (plus de) deux chaussettes.*

Correction de l'exercice 1 ▲

La seule chose à voir est que pour une division euclidienne le reste doit être plus petit que le quotient. Donc les divisions euclidiennes s'écrivent : $96842 = 256 \times 378 + 74$ et $96842 = 258 \times 375 + 92$.

Correction de l'exercice 2 ▲

Il suffit de constater que pour 4 nombres consécutifs il y a nécessairement : un diviseur de 2, un diviseur de 3, un diviseur de 4 (tous distincts). Donc le produit de 4 nombres consécutifs est divisible par $2 \times 3 \times 4 = 24$.

Correction de l'exercice 3 ▲

Ecrire $n = p^2 + q^2$ et étudier le reste de la division euclidienne de n par 4 en distinguant les différents cas de parité de p et q.

Correction de l'exercice 4 ▲

Raisonnons modulo 8 :

$$7 \equiv -1 \pmod 8.$$

Donc

$$7^n + 1 \equiv (-1)^n + 1 \pmod 8.$$

Le reste de la division euclidienne de $7^n + 1$ par 8 est donc $(-1)^n + 1$ donc Si n est impair alors $7^n + 1$ est divisible par 8. Et si n est pair $7^n + 1$ n'est pas divisible par 8.

Correction de l'exercice 5 ▲

Il sagit de calculer 100^{1000} modulo 13. Tout d'abord $100 \equiv 9 \pmod{13}$ donc $100^{1000} \equiv 9^{1000} \pmod{13}$. Or $9^2 \equiv 81 \equiv 3 \pmod{13}$, $9^3 \equiv 9^2.9 \equiv 3.9 \equiv 1 \pmod{13}$, Or $9^4 \equiv 9^3.9 \equiv 9 \pmod{13}$, $9^5 \equiv 9^4.9 \equiv 9.9 \equiv 3 \pmod{13}$. Donc $100^{1000} \equiv 9^{1000} \equiv 9^{3.333+1} \equiv (9^3)^{333}.9 \equiv 1^{333}.9 \equiv 9 \pmod{13}$.

Correction de l'exercice 6 ▲

1. Soit n un nombre impair, alors il s'écrit $n = 2p+1$ avec $p \in \mathbb{N}$. Maintenant $n^2 = (2p+1)^2 = 4p^2 + 4p + 1 = 4p(p+1) + 1$. Donc $n^2 \equiv 1 \pmod 8$.
2. Si n est pair alors il existe $p \in \mathbb{N}$ tel que $n = 2p$. Et $n^2 = 4p^2$. Si p est pair alors p^2 est pair et donc $n^2 = 4p^2$ est divisible par 8, donc $n^2 \equiv 0 \pmod 8$. Si p est impair alors p^2 est impair et donc $n^2 = 4p^2$ est divisible par 4 mais pas par 8, donc $n^2 \equiv 4 \pmod 8$.
3. Comme a est impair alors d'après la première question $a^2 \equiv 1 \pmod 8$, et de même $c^2 \equiv 1 \pmod 8$, $c^2 \equiv 1 \pmod 8$. Donc $a^2 + b^2 + c^2 \equiv 1 + 1 + 1 \equiv 3 \pmod 8$. Pour l'autre reste, écrivons $a = 2p+1$ et $b = 2q+1$, $c = 2r+1$, alors $2ab = 2(2p+1)(2q+1) = 8pq + 4(p+q) + 2$. Alors $2(ab+bc+ca) = 8pq + 8qr + 8pr + 8(p+q+r) + 6$, donc $2(ab+bc+ca) \equiv 6 \pmod 8$.
4. Montrons par l'absurde que le nombre $a^2 + b^2 + c^2$ n'est pas le carré d'un nombre entier. Supposons qu'il existe $n \in \mathbb{N}$ tel que $a^2 + b^2 + c^2 = n^2$. Nous savons que $a^2 + b^2 + c^2 \equiv 3 \pmod 8$. Si n est impair alors $n^2 \equiv 1 \pmod 8$ et si n est pair alors $n^2 \equiv 0 \pmod 8$ ou $n^2 \equiv 4 \pmod 8$. Dans tous les cas n^2 n'est pas congru à 3 modulo 8. Donc il y a une contradiction. La conclusion est que l'hypothèse de départ est fausse donc $a^2 + b^2 + c^2$ n'est pas un carré. Le même type de raisonnement est valide pour $2(ab+bc+ca)$.

   Pour $ab+bc+ca$ l'argument est similaire : d'une part $2(ab+bc+ca) \equiv 6 \pmod 8$ et d'autre part si, par l'absurde, on suppose $ab+bc+ca = n^2$ alors selon la parité de n nous avons $2(ab+bc+ca) \equiv 2n^2 \equiv 2 \pmod 8$ ou à $0 \pmod 8$. Dans les deux cas cela aboutit à une contradiction. Nous avons montrer que $ab+bc+ca$ n'est pas un carré.

Correction de l'exercice 7 ▲

Il s'agit ici d'utiliser la décomposition des nombres en facteurs premiers.

1. $126 = 2.3^2.7$ et $230 = 2.5.23$ donc le pgcd de 126 et 230 est 2.

2. $390 = 2.3.5.13$, $720 = 2^4.3^2.5$, $450 = 2.3^2.5^2$ et donc le pgcd de ces trois nombres est $2.3.5 = 30$.
3. $\text{pgcd}(180, 606, 750) = 6$.

Correction de l'exercice 8 ▲

Soient a, b deux entiers de pgcd 18 et de somme 360. Soit a', b' tel que $a = 18a'$ et $b = 18b'$. Alors a' et b' sont premiers entre eux, et leur somme est $360/18 = 20$.
Nous pouvons facilement énumérer tous les couples d'entiers naturels (a', b') $(a' \leqslant b')$ qui vérifient cette condition, ce sont les couples :

$$(1, 19), (3, 17), (7, 13), (9, 11).$$

Pour obtenir les couples (a, b) recherchés $(a \leqslant b)$, il suffit de multiplier les couples précédents par 18 :

$$(18, 342), (54, 306), (126, 234), (162, 198).$$

Correction de l'exercice 9 ▲

1. $\text{pgcd}(18480, 9828) = 84$;
2. $25 \times 18480 + (-47) \times 9828 = 84$.

Correction de l'exercice 10 ▲

1. $a = 9b + 10$.
2. Calculons le pgcd par l'algorithme d'Euclide. $a = 9b + 10$, $b = 12345678 \times 10 + 9$, $10 = 1 \times 9 + 1$. Donc le pgcd vaut 1 ;
3. Nous reprenons les équations précédentes en partant de la fin : $1 = 10 - 9$, puis nous remplaçons 9 grâce à la deuxième équation de l'algorithme d'Euclide : $1 = 10 - (b - 12345678 \times 10) = -b + 1234679 \times 10$. Maintenant nous remplaçons 10 grâce à la première équation : $1 = -b + 12345679(a - 9b) = 12345679a - 111111112b$.

Correction de l'exercice 11 ▲

En divisant par 45 (qui est le pgcd de $1665, 1035, 45$) nous obtenons l'équation équivalente :

$$37x + 23y = 1 \qquad (E)$$

Comme le pgcd de 37 et 23 est 1, alors d'après le théorème de Bézout cette équation (E) a des solutions.
L'algorithme d'Euclide pour le calcul du pgcd de 37 et 23 fourni les coefficients de Bézout : $37 \times 5 + 23 \times (-8) = 1$. Une solution particulière de (E) est donc $(x_0, y_0) = (5, -8)$.
Nous allons maintenant trouver l'expression générale pour les solutions de l'équation (E). Soient (x, y) une solution de l'équation $37x + 23y = 1$. Comme (x_0, y_0) est aussi solution, nous avons $37x_0 + 23y_0 = 1$. Faisons la différence de ces deux égalités pour obtenir $37(x - x_0) + 23(y - y_0) = 0$. Autrement dit

$$37(x - x_0) = -23(y - y_0) \quad (*)$$

On en déduit que $37 | 23(y - y_0)$, or $\text{pgcd}(23, 37) = 1$ donc par le lemme de Gauss, $37 | (y - y_0)$. (C'est ici qu'il est important d'avoir divisé par 45 dès le début !) Cela nous permet d'écrire $y - y_0 = 37k$ pour un $k \in \mathbb{Z}$.
Repartant de l'égalité $(*)$: nous obtenons $37(x - x_0) = -23 \times 37 \times k$. Ce qui donne $x - x_0 = -23k$. Donc si (x, y) est solution de (E) alors elle est de la forme : $(x, y) = (x_0 - 23k, y_0 + 37k)$, avec $k \in \mathbb{Z}$.
Réciproquement pour chaque $k \in \mathbb{Z}$, si (x, y) est de cette forme alors c'est une solution de (E) (vérifiez-le !).
Conclusion : les solutions sont

$$\{(5 - 23k, -8 + 37k) \mid k \in \mathbb{Z}\}.$$

Correction de l'exercice 12 ▲

Écrivons la décomposition de $15! = 1.2.3.4\ldots15$ en facteurs premiers. $15! = 2^{11}.3^6.5^3.7^2.11.13$. Un diviseur de $15!$ s'écrit $d = 2^{\alpha}.3^{\beta}.5^{\gamma}.7^{\delta}.11^{\varepsilon}.13^{\eta}$ avec $0 \leq \alpha \leq 11$, $0 \leq \beta \leq 6$, $0 \leq \gamma \leq 3$, $0 \leq \delta \leq 2$, $0 \leq \varepsilon \leq 1$, $0 \leq \eta \leq 1$. De plus tout nombre d de cette forme est un diviseur de $15!$. Le nombre de diviseurs est donc $(11+1)(6+1)(3+1)(2+1)(1+1)(1+1) = 4032$.

Correction de l'exercice 13 ▲

Soit a et b des entiers premiers entre eux. Raisonnons par l'absurde et supposons que ab et $a+b$ ne sont pas premiers entre eux. Il existe alors un nombre premier divisant ab et $a+b$. Par le lemme d'Euclide comme $p|ab$ alors $p|a$ ou $p|b$. Par exemple supposons que $p|a$. Comme $p|a+b$ alors p divise aussi $(a+b)-a$, donc $p|b$. δ ne divise pas b cela implique que δ et b sont premiers entre eux.
D'après le lemme de Gauss, comme δ divise ab et δ premier avec b alors δ divise a. Donc p est un facteur premier de a et de b ce qui est absurde.

Correction de l'exercice 14 ▲

1. Nous savons que
$$x^b - 1 = (x-1)(x^{b-1} + \cdots + x + 1),$$
pour $x = 2^a$ nous obtenons :
$$2^{ab} - 1 = (2^a)^b - 1 = (2^a - 1)\left(2^{a(b-1)} + \cdots + 2^a + 1\right).$$
Donc $(2^a - 1)|(2^{ab} - 1)$.
2. Montrons la contraposée. Supposons que p ne soit pas premier. Donc $p = ab$ avec $1 < p, q < a$. Par la question précédente $2^a - 1$ divise $2^p - 1$ (et $1 < 2^a - 1 < 2^p - 1$). Donc $2^p - 1$ n'est pas un nombre premier.
3. Nous supposons $a \geq b$. Nous allons montrer que faire l'algorithme d'Euclide pour le couple $(2^a - 1, 2^b - 1)$ revient à faire l'algorithme d'Euclide pour (a,b). Tout d'abord rappellons la formule qui est à la base de l'algorithme d'Euclide : $\text{pgcd}(a,b) = \text{pgcd}(a-b,b)$. Appliqué à $2^a - 1$ et $2^b - 1$ cela donne directement $\text{pgcd}(2^a - 1, 2^b - 1) = \text{pgcd}(2^a - 2^b, 2^b - 1)$. Mais $2^a - 2^b = 2^b(2^{a-b} - 1)$ d'où $\text{pgcd}(2^a - 1, 2^b - 1) = \text{pgcd}(2^b(2^{a-b} - 1), 2^b - 1) = \text{pgcd}(2^{a-b} - 1, 2^b - 1)$. La dernière égalité vient du fait 2^b et $2^b - 1$ sont premiers entre eux (deux entiers consécutifs sont toujours premiers entre eux).
Nous avons montrer : $\text{pgcd}(2^a - 1, 2^b - 1) = \text{pgcd}(2^{a-b} - 1, 2^b - 1)$. Cette formule est à mettre en parallèle de $\text{pgcd}(a,b) = \text{pgcd}(a-b,b)$. En itérant cette formule nous obtenons que si $a = bq + r$ alors : $\text{pgcd}(2^a - 1, 2^b - 1) = \text{pgcd}(2^{a-bq} - 1, 2^b - 1) = \text{pgcd}(2^r - 1, 2^b - 1)$ à comparer avec $\text{pgcd}(a,b) = \text{pgcd}(a - bq, b) = \text{pgcd}(r,b)$. Nous avons notre première étape de l'algorithme d'Euclide. En itérant l'algorithme d'Euclide pour (a,b), nous nous arêtons au dernier reste non nul : $\text{pgcd}(a,b) = \text{pgcd}(b,r) = \cdots = \text{pgcd}(r_n, 0) = r_n$. Ce qui va donner pour nous $\text{pgcd}(2^a - 1, 2^b - 1) = \text{pgcd}(2^b - 1, 2^r - 1) = \cdots = \text{pgcd}(2^{r_n} - 1, 2^0 - 1) = 2^{r_n} - 1$.
Bilan : $\text{pgcd}(2^a - 1, 2^b - 1) = 2^{\text{pgcd}(a,b)} - 1$.

Correction de l'exercice 15 ▲

1. Supposons que $a^n + 1$ est premier. Nous allons montrer la contraposée. Supposons que n n'est pas de la forme 2^k, c'est-à-dire que $n = p \times q$ avec p un nombre premier > 2 et $q \in \mathbb{N}$. Nous utilisons la formule
$$x^p + 1 = (x+1)(1 - x + x^2 - x^3 + \ldots + x^{p-1})$$
avec $x = a^q$:
$$a^n + 1 = a^{pq} + 1 = (a^q)^p + 1 = (a^q + 1)(1 - a^q + (a^q)^2 + \cdots + (a^q)^{p-1}).$$

Donc a^q+1 divise a^n+1 et comme $1<a^q+1<a^n+1$ alors a^n+1 n'est pas premier. Par contraposition si a^n+1 est premier alor $n=2^k$.

2. Cette conjecture est fausse, mais pas facile à vérifier sans une bonne calculette ! En effet pour $n=5$ nous obtenons :
$$2^{2^5}+1=4294967297=641\times 6700417.$$

Correction de l'exercice 16 ▲

1. Étant donné $0<i<p$, nous avons
$$C_p^i=\frac{p!}{i!(p-i)!}=\frac{p(p-1)(p-2)\dots(p-(i+1))}{i!}$$
Comme C_p^i est un entier alors $i!$ divise $p(p-1)\dots(p-(i+1))$. Mais $i!$ et p sont premiers entre eux (en utilisant l'hypothèse $0<i<p$). Donc d'après le théorème de Gauss : $i!$ divise $(p-1)\dots(p-(i+1))$, autrement dit il existe $k\in\mathbb{Z}$ tel que $ki!=(p-1)\dots(p-(i+1))$. Maintenant nous avons $C_p^i=pk$ donc p divise C_p^i.
2. Il s'agit de montrer le petit théorème de Fermat : pour p premier et $a\in\mathbb{N}^*$, alors $a^p\equiv a \pmod p$. Fixons p. Soit l'assertion
$$(\mathscr{H}_a)\quad a^p\equiv a \pmod p.$$
Pour $a=1$ cette assertion est vraie ! Étant donné $a\geq 1$ supposons que $\mathscr{H}_a$ soit vraie. Alors
$$(a+1)^p=\sum_{i=0}^{p}C_p^i a^i.$$
Mais d'après la question précédente pour $0<i<p$, p divise C_p^i. En termes de modulo nous obtenons :
$$(a+1)^p\equiv C_p^0a^0+C_p^pa^p\equiv 1+a^p \pmod p.$$
Par l'hypothèse de récurrence nous savons que $a^p\equiv a \pmod p$, donc
$$(a+1)^p\equiv a+1 \pmod p.$$
Nous venons de prouver que $\mathscr{H}_{a+1}$ est vraie. Par le principe de récurrence alors quelque soit $a\in\mathbb{N}^*$ nous avons :
$$a^p\equiv a \pmod p.$$

Correction de l'exercice 17 ▲

1. Fixons n et montrons la récurrence sur $k\in\mathbb{N}$. La formule est vraie pour $k=0$. Supposons la formule vraie au rang k. Alors
$$\begin{aligned}(2^{2^n}-1)\times\prod_{i=0}^{k}(2^{2^{n+i}}+1)&=(2^{2^n}-1)\times\prod_{i=0}^{k-1}(2^{2^{n+i}}+1)\times(2^{2^{n+k}}+1)\\&=(2^{2^{n+k}}-1)\times(2^{2^{n+k}}+1)=(2^{2^{n+k}})^2-1=2^{2^{n+k+1}}-1.\end{aligned}$$
Nous avons utiliser l'hypothèse de récurrence dans ces égalités. Nous avons ainsi montrer la formule au rang $k+1$. Et donc par le principe de récurrence elle est vraie.

2. Écrivons $m = n+k$, alors l'égalité précédente devient :

$$F_m + 2 = (2^{2^n} - 1) \times \prod_{i=n}^{m-1} F_i.$$

Soit encore :

$$F_n \times (2^{2^n} - 1) \times \prod_{i=n+1}^{m-1} F_i \quad - \quad F_m = 2.$$

Si d est un diviseur de F_n et F_m alors d divise 2 (ou alors on peut utiliser le théorème de Bézout). En conséquent $d = 1$ ou $d = 2$. Mais F_n est impair donc $d = 1$. Nous avons montrer que tous diviseurs de F_n et F_m est 1, cela signifie que F_n et F_m sont premiers entre eux.

3. Supposons qu'il y a un nombre fini de nombres premiers. Nous les notons alors $\{p_1, \ldots, p_N\}$. Prenons alors $N+1$ nombres de la famille F_i, par exemple $\{F_1, \ldots, F_{N+1}\}$. Chaque F_i, $i = 1, \ldots, N+1$ est divisible par (au moins) un facteur premier p_j, $j = 1, \ldots, N$. Nous avons $N+1$ nombres F_i et seulement N facteurs premiers p_j. Donc par le principe des tiroirs il existe deux nombres distincts F_k et $F_{k'}$ (avec $1 \leq k, k' \leq N+1$) qui ont un facteur premier en commun. En conséquent F_k et $F_{k'}$ ne sont pas premiers entre eux. Ce qui contredit la question précédente. Il existe donc une infinité de nombres premiers.

Correction de l'exercice 18 ▲

1. X est non vide car, par exemple pour $k = 2$, $4k+3 = 11$ est premier.
2. $(4k+1)(4\ell+1) = 16k\ell + 4(k+\ell) + 1 = 4(4k\ell + k + \ell) + 1$. Si l'on note l'entier $k' = 4k\ell + k + \ell$ alors $(4k+1)(4\ell+1) = 4k' + 1$, ce qui est bien de la forme voulue.
3. Remarquons que 2 est le seul nombre premier pair, les autres sont de la forme $4k+1$ ou $4k+3$. Ici a n'est pas divisible par 2, supposons –par l'absurde– que a n'a pas de diviseur de la forme $4k+3$, alors tous les diviseurs de a sont de la forme $4k+1$. C'est-à-dire que a s'écrit comme produit de nombre de la forme $4k+1$, et par la question précédente a peut s'écrire $a = 4k' + 1$. Donc $a \equiv 1 \pmod 4$. Mais comme $a = 4p_1 p_2 \ldots p_n - 1$, $a \equiv -1 \equiv 3 \pmod 4$. Nous obtenons une contradiction. Donc a admet une diviseur premier p de la forme $p = 4\ell + 3$.
4. Dans l'ensemble $X = \{p_1, \ldots, p_n\}$ il y a tous les nombres premiers de la formes $4k+3$. Le nombre p est premier et s'écrit $p = 4\ell + 3$ donc p est un élément de X, donc il existe $i \in \{1, \ldots, n\}$ tel que $p = p_i$. Raisonnons modulo $p = p_i$: $a \equiv 0 \pmod p$ car p divise a. D'autre part $a = 4p_1 \ldots p_n - 1$ donc $a \equiv -1 \pmod p$. (car p_i divise $p_1 \ldots p_n$). Nous obtenons une contradiction, donc X est infini : il existe une infinité de nombre premier de la forme $4k+3$. Petite remarque, tous les nombres de la forme $4k+3$ ne sont pas des nombres premiers, par exemple pour $k = 3$, $4k+3 = 15$ n'est pas premier.

Logique, ensembles, raisonnements

1 Logique

Exercice 1

Compléter les pointillés par le connecteur logique qui s'impose : $\Leftrightarrow$, $\Leftarrow$, $\Rightarrow$.

1. $x \in \mathbb{R}$ $x^2 = 4$ ..... $x = 2$;
2. $z \in \mathbb{C}$ $z = \bar{z}$ ..... $z \in \mathbb{R}$;
3. $x \in \mathbb{R}$ $x = \pi$ ..... $e^{2ix} = 1$.

Correction ▼ Vidéo ■ [000108]

Exercice 2

Soient les quatre assertions suivantes :

$$(a)\ \exists x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad x+y>0 \quad ; \quad (b)\ \forall x \in \mathbb{R} \quad \exists y \in \mathbb{R} \quad x+y>0\ ;$$

$$(c)\ \forall x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad x+y>0 \quad ; \quad (d)\ \exists x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad y^2>x.$$

1. Les assertions a, b, c, d sont-elles vraies ou fausses ?
2. Donner leur négation.

Indication ▼ Correction ▼ Vidéo ■ [000106]

Exercice 3

Dans $\mathbb{R}^2$, on définit les ensembles $F_1 = \{(x,y) \in \mathbb{R}^2,\ y \leq 0\}$ et $F_2 = \{(x,y) \in \mathbb{R}^2,\ xy \geq 1,\ x \geq 0\}$. On note M_1M_2 la distance usuelle entre deux points M_1 et M_2 de $\mathbb{R}^2$. Évaluer les propositions suivantes :

1. $\forall \varepsilon \in]0,+\infty[\quad \exists M_1 \in F_1 \quad \exists M_2 \in F_2 \quad M_1M_2 < \varepsilon$
2. $\exists M_1 \in F_1 \quad \exists M_2 \in F_2 \quad \forall \varepsilon \in]0,+\infty[\quad M_1M_2 < \varepsilon$
3. $\exists \varepsilon \in]0,+\infty[\quad \forall M_1 \in F_1 \quad \forall M_2 \in F_2 \quad M_1M_2 < \varepsilon$
4. $\forall M_1 \in F_1 \quad \forall M_2 \in F_2 \quad \exists \varepsilon \in]0,+\infty[\quad M_1M_2 < \varepsilon$

Quand elles sont fausses, donner leur négation.

Indication ▼ Correction ▼ Vidéo ■ [000109]

Exercice 4

Nier la proposition : "tous les habitants de la rue du Havre qui ont les yeux bleus gagneront au loto et prendront leur retraite avant 50 ans".

Correction ▼ Vidéo ■ [000110]

Exercice 5

Nier les assertions suivantes :

1. tout triangle rectangle possède un angle droit ;
2. dans toutes les écuries, tous les chevaux sont noirs ;

3. pour tout entier x, il existe un entier y tel que, pour tout entier z, la relation $z < x$ implique le relation $z < x+1$;
4. $\forall \varepsilon > 0 \quad \exists \alpha > 0 \qquad (|x-7/5| < \alpha \Rightarrow |5x-7| < \varepsilon)$.

Correction ▼ Vidéo ■ [000112]

Exercice 6

Soient f, g deux fonctions de $\mathbb{R}$ dans $\mathbb{R}$. Traduire en termes de quantificateurs les expressions suivantes :

1. f est majorée ;
2. f est bornée ;
3. f est paire ;
4. f est impaire ;
5. f ne s'annule jamais ;
6. f est périodique ;
7. f est croissante ;
8. f est strictement décroissante ;
9. f n'est pas la fonction nulle ;
10. f n'a jamais les mêmes valeurs en deux points distincts ;
11. f atteint toutes les valeurs de $\mathbb{N}$;
12. f est inférieure à g ;
13. f n'est pas inférieure à g.

Correction ▼ Vidéo ■ [000120]

Exercice 7

Soit f une application de $\mathbb{R}$ dans $\mathbb{R}$. Nier, de la manière la plus précise possible, les énoncés qui suivent :

1. Pour tout $x \in \mathbb{R}$ $f(x) \leq 1$.
2. L'application f est croissante.
3. L'application f est croissante et positive.
4. Il existe $x \in \mathbb{R}^+$ tel que $f(x) \leq 0$.
5. Il existe $x \in \mathbb{R}$ tel que quel que soit $y \in \mathbb{R}$, si $x < y$ alors $f(x) > f(y)$.

On ne demande pas de démontrer quoi que ce soit, juste d'écrire le contraire d'un énoncé.

Correction ▼ Vidéo ■ [000107]

Exercice 8

Montrer que

$$\forall \varepsilon > 0 \quad \exists N \in \mathbb{N} \text{ tel que } \left(n \geq N \Rightarrow 2 - \varepsilon < \frac{2n+1}{n+2} < 2 + \varepsilon\right).$$

Indication ▼ Correction ▼ Vidéo ■ [000119]

2 Ensembles

Exercice 9

Soit A, B deux ensembles, montrer $\complement(A \cup B) = \complement A \cap \complement B$ et $\complement(A \cap B) = \complement A \cup \complement B$.

Indication ▼ Correction ▼ Vidéo ■ [000123]

Exercice 10

Montrer par contraposition les assertions suivantes, E étant un ensemble :

1. $\forall A,B \in \mathscr{P}(E) \quad (A \cap B = A \cup B) \Rightarrow A = B$,
2. $\forall A,B,C \in \mathscr{P}(E) \quad (A \cap B = A \cap C \text{ et } A \cup B = A \cup C) \Rightarrow B = C$.

Correction ▼ Vidéo ■ [000122]

Exercice 11

Soient E et F deux ensembles, $f : E \to F$. Démontrer que :
$\forall A,B \in \mathscr{P}(E) \quad (A \subset B) \Rightarrow (f(A) \subset f(B))$,
$\forall A,B \in \mathscr{P}(E) \quad f(A \cap B) \subset f(A) \cap f(B)$,
$\forall A,B \in \mathscr{P}(E) \quad f(A \cup B) = f(A) \cup f(B)$,
$\forall A,B \in \mathscr{P}(F) \quad f^{-1}(A \cup B) = f^{-1}(A) \cup f^{-1}(B)$,
$\forall A \in \mathscr{P}(F) \quad f^{-1}(F \setminus A) = E \setminus f^{-1}(A)$.

Correction ▼ Vidéo ■ [000124]

Exercice 12

Montrez que chacun des ensembles suivants est un intervalle que vous calculerez.

$$I = \bigcap_{n=1}^{+\infty} \left[-\frac{1}{n}, 2+\frac{1}{n}\right[\quad \text{et} \quad J = \bigcup_{n=2}^{+\infty} \left[1+\frac{1}{n}, n\right]$$

Correction ▼ Vidéo ■ [000137]

3 Absurde et contraposée

Exercice 13

Soit $(f_n)_{n\in\mathbb{N}}$ une suite d'applications de l'ensemble $\mathbb{N}$ dans lui-même. On définit une application f de $\mathbb{N}$ dans $\mathbb{N}$ en posant $f(n) = f_n(n) + 1$. Démontrer qu'il n'existe aucun $p \in \mathbb{N}$ tel que $f = f_p$.

Indication ▼ Correction ▼ Vidéo ■ [000150]

Exercice 14

1. Soit $p_1, p_2, \ldots, p_r$, r nombres premiers. Montrer que l'entier $N = p_1 p_2 \ldots p_r + 1$ n'est divisible par aucun des entiers p_i.
2. Utiliser la question précédente pour montrer par l'absurde qu'il existe une infinité de nombres premiers.

Indication ▼ Correction ▼ Vidéo ■ [000151]

4 Récurrence

Exercice 15

Montrer :

1. $\displaystyle\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \quad \forall n \in \mathbb{N}^*$.
2. $\displaystyle\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6} \quad \forall n \in \mathbb{N}^*$.

Correction ▼ Vidéo ■ [000153]

Exercice 16

Soit X un ensemble. Pour $f \in \mathscr{F}(X,X)$, on définit $f^0 = id$ et par récurrence pour $n \in \mathbb{N}$ $f^{n+1} = f^n \circ f$.

1. Montrer que $\forall n \in \mathbb{N}$ $f^{n+1} = f \circ f^n$.

2. Montrer que si f est bijective alors $\forall n \in \mathbb{N}\ (f^{-1})^n = (f^n)^{-1}$.

Indication ▼ Correction ▼ Vidéo ■ [000157]

Exercice 17

Soit la suite $(x_n)_{n\in\mathbb{N}}$ définie par $x_0 = 4$ et $x_{n+1} = \dfrac{2x_n^2 - 3}{x_n + 2}$.

1. Montrer que : $\forall n \in \mathbb{N} \quad x_n > 3$.
2. Montrer que : $\forall n \in \mathbb{N} \quad x_{n+1} - 3 > \frac{3}{2}(x_n - 3)$.
3. Montrer que : $\forall n \in \mathbb{N} \quad x_n \geqslant \left(\frac{3}{2}\right)^n + 3$.
4. La suite $(x_n)_{n\in\mathbb{N}}$ est-elle convergente ?

Indication ▼ Correction ▼ Vidéo ■ [000155]

Indication pour l'exercice 2 ▲

Attention : la négation d'une inégalité stricte est une inégalité large (et réciproquement).

Indication pour l'exercice 3 ▲

Faire un dessin de F_1 et de F_2. Essayer de voir si la difficulté pour réaliser les assertions vient de ε "petit" (c'est-à-dire proche de 0) ou de ε "grand" (quand il tend vers $+\infty$).

Indication pour l'exercice 8 ▲

En fait, on a toujours : $\frac{2n+1}{n+2} \leq 2$. Puis chercher une condition sur n pour que l'inégalité

$$2-\varepsilon < \frac{2n+1}{n+2}$$

soit vraie.

Indication pour l'exercice 9 ▲

Il est plus facile de raisonner en prenant un élément $x \in E$. Par exemple, soit F, G des sous-ensembles de E. Montrer que $F \subset G$ revient à montrer que pour tout $x \in F$ alors $x \in G$. Et montrer $F = G$ est équivalent à $x \in F$ si et seulement si $x \in G$, et ce pour tout x de E. Remarque : pour montrer $F = G$ on peut aussi montrer $F \subset G$ puis $G \subset F$.
Enfin, se rappeler que $x \in \complement F$ si et seulement si $x \notin F$.

Indication pour l'exercice 13 ▲

Par l'absurde, supposer qu'il existe $p \in \mathbb{N}$ tel que $f = f_p$. Puis pour un tel p, évaluer f et f_p en une valeur bien choisie.

Indication pour l'exercice 14 ▲

Pour la première question vous pouvez raisonner par contraposition ou par l'absurde.

Indication pour l'exercice 16 ▲

Pour les deux questions, travailler par récurrence.

Indication pour l'exercice 17 ▲

1. Récurrence : calculer $x_{n+1} - 3$.
2. Calculer $x_{n+1} - 3 - \frac{3}{2}(x_n - 3)$.
3. Récurrence.

Correction de l'exercice 1 ▲

1. $\Leftarrow$
2. $\Leftrightarrow$
3. $\Rightarrow$

Correction de l'exercice 2 ▲

1. (a) est fausse. Car sa négation qui est $\forall x \in \mathbb{R}\ \exists y \in \mathbb{R} \quad x+y \leq 0$ est vraie. Étant donné $x \in \mathbb{R}$ il existe toujours un $y \in \mathbb{R}$ tel que $x+y \leq 0$, par exemple on peut prendre $y = -(x+1)$ et alors $x+y = x-x-1 = -1 \leq 0$.
2. (b) est vraie, pour un x donné, on peut prendre (par exemple) $y = -x+1$ et alors $x+y = 1 > 0$. La négation de (b) est $\exists x \in \mathbb{R}\ \forall y \in \mathbb{R} \quad x+y \leq 0$.
3. (c) : $\forall x \in \mathbb{R}\ \forall y \in \mathbb{R} \quad x+y > 0$ est fausse, par exemple $x = -1$, $y = 0$. La négation est $\exists x \in \mathbb{R}\ \exists y \in \mathbb{R}\ x+y \leq 0$.
4. (d) est vraie, on peut prendre $x = -1$. La négation est : $\forall x \in \mathbb{R}\ \exists y \in \mathbb{R} \quad y^2 \leq x$.

Correction de l'exercice 3 ▲

1. Cette proposition est vraie. En effet soit $\varepsilon > 0$, définissons $M_1 = (\frac{2}{\varepsilon}, 0) \in F_1$ et $M_2 = (\frac{2}{\varepsilon}, \frac{\varepsilon}{2}) \in F_2$, alors $M_1M_2 = \frac{\varepsilon}{2} < \varepsilon$. Ceci étant vrai quelque soit $\varepsilon > 0$ la proposition est donc démontrée.
2. Soit deux points fixés M_1, M_2 vérifiant cette proposition, la distance $d = M_1M_2$ est aussi petite que l'on veut donc elle est nulle, donc $M_1 = M_2$; or les ensembles F_1 et F_2 sont disjoints. Donc la proposition est fausse. La négation de cette proposition est :

$$\forall M_1 \in F_1\ \ \forall M_2 \in F_2 \quad \exists \varepsilon \in]0, +\infty[\qquad M_1M_2 \geqslant \varepsilon$$

   et cela exprime le fait que les ensembles F_1 et F_2 sont disjoints.
3. Celle ci est également fausse, en effet supposons qu'elle soit vraie, soit alors ε correspondant à cette proposition. Soit $M_1 = (\varepsilon+2, 0)$ et $M_2 = (1,1)$, on a $M_1M_2 > \varepsilon+1$ ce qui est absurde. La négation est :

$$\forall \varepsilon \in]0, +\infty[\quad \exists M_1 \in F_1\ \ \exists M_2 \in F_2 \qquad M_1M_2 \geqslant \varepsilon$$

   C'est-à-dire que l'on peut trouver deux points aussi éloignés l'un de l'autre que l'on veut.
4. Cette proposition est vraie, il suffit de choisir $\varepsilon = M_1M_2 + 1$. Elle signifie que la distance entre deux points donnés est un nombre fini !

Correction de l'exercice 4 ▲

"Il existe un habitant de la rue du Havre qui a les yeux bleus, qui ne gagnera pas au loto ou qui prendra sa retraite après 50 ans."

Correction de l'exercice 5 ▲

1. "Il existe un triangle rectangle qui n'a pas d'angle droit." Bien sûr cette dernière phrase est fausse !
2. "Il existe une écurie dans laquelle il y a (au moins) un cheval dont la couleur n'est pas noire."
3. Sachant que la proposition en langage mathématique s'écrit

$$\forall x \in \mathbb{Z}\ \ \exists y \in \mathbb{Z}\ \ \forall z \in \mathbb{Z} \quad (z < x \Rightarrow z < x+1),$$

   la négation est

$$\exists x \in \mathbb{Z}\ \ \forall y \in \mathbb{Z}\ \ \exists z \in \mathbb{Z} \quad (z < x \text{ et } z \geq x+1).$$

4. $\exists \varepsilon > 0 \ \ \forall \alpha > 0 \quad (|x - 7/5| < \alpha \text{ et } |5x - 7| \geq \varepsilon)$.

Correction de l'exercice 6 ▲

1. $\exists M \in \mathbb{R} \quad \forall x \in \mathbb{R} \qquad f(x) \leq M$;
2. $\exists M \in \mathbb{R} \quad \exists m \in \mathbb{R} \quad \forall x \in \mathbb{R} \qquad m \leq f(x) \leq M$;
3. $\forall x \in \mathbb{R} \qquad f(x) = f(-x)$;
4. $\forall x \in \mathbb{R} \qquad f(x) = -f(-x)$;
5. $\forall x \in \mathbb{R} \qquad f(x) \neq 0$;
6. $\exists a \in \mathbb{R}^* \quad \forall x \in \mathbb{R} \qquad f(x+a) = f(x)$;
7. $\forall (x,y) \in \mathbb{R}^2 \qquad (x \leq y \Rightarrow f(x) \leq f(y))$;
8. $\forall (x,y) \in \mathbb{R}^2 \qquad (x < y \Rightarrow f(x) > f(y))$;
9. $\exists x \in \mathbb{R} \qquad f(x) \neq 0$;
10. $\forall (x,y) \in \mathbb{R}^2 \qquad (x \neq y \Rightarrow f(x) \neq f(y))$;
11. $\forall n \in \mathbb{N} \quad \exists x \in \mathbb{R} \qquad f(x) = n$;
12. $\forall x \in \mathbb{R} \qquad f(x) \leq g(x)$;
13. $\exists x \in \mathbb{R} \qquad f(x) > g(x)$.

Correction de l'exercice 7 ▲

Dans ce corrigé, nous donnons une justification, ce qui n'était pas demandé.

1. Cette assertion se décompose de la manière suivante : (Pour tout $x \in \mathbb{R}$) ($f(x) \leq 1$). La négation de "(Pour tout $x \in \mathbb{R}$)" est "Il existe $x \in \mathbb{R}$" et la négation de "($f(x) \leq 1$)" est $f(x) > 1$. Donc la négation de l'assertion complète est : "Il existe $x \in \mathbb{R}, f(x) > 1$".
2. Rappelons comment se traduit l'assertion "L'application f est croissante" : "pour tout couple de réels (x_1, x_2), si $x_1 \leq x_2$ alors $f(x_1) \leq f(x_2)$". Cela se décompose en : "(pour tout couple de réels x_1 et x_2) ($x_1 \leq x_2$ implique $f(x_1) \leq f(x_2)$)". La négation de la première partie est : "(il existe un couple de réels (x_1, x_2))" et la négation de la deuxième partie est : "($x_1 \leq x_2$ et $f(x_1) > f(x_2)$)". Donc la négation de l'assertion complète est : "Il existe $x_1 \in \mathbb{R}$ et $x_2 \in \mathbb{R}$ tels que $x_1 \leq x_2$ et $f(x_1) > f(x_2)$".
3. La négation est : "l'application f n'est pas croissante ou n'est pas positive". On a déjà traduit "l'application f n'est pas croissante", traduisons "l'application f n'est pas positive" : "il existe $x \in \mathbb{R}, f(x) < 0$". Donc la négation de l'assertion complète est : " Il existe $x_1 \in \mathbb{R}$ et $x_2 \in \mathbb{R}$ tels que $x_1 < x_2$ et $f(x_1) \geq f(x_2)$, ou il existe $x \in \mathbb{R}, f(x) < 0$".
4. Cette assertion se décompose de la manière suivante : "(Il existe $x \in \mathbb{R}^+$) ($f(x) \leq 0$)". La négation de la première partie est : "(pour tout $x \in \mathbb{R}^+$)", et celle de la seconde est :"($f(x) > 0$)". Donc la négation de l'assertion complète est : "Pour tout $x \in \mathbb{R}^+$, $f(x) > 0$".
5. Cette assertion se décompose de la manière suivante : "$(\exists x \in \mathbb{R})(\forall y \in \mathbb{R})(x < y \Rightarrow f(x) > f(y))$". La négation de la première partie est "$(\forall x \in \mathbb{R})$", celle de la seconde est "$(\exists y \in \mathbb{R})$", et celle de la troisième est "($x < y$ et $f(x) \leq f(y)$)". Donc la négation de l'assertion complète est : " $\forall x \in \mathbb{R}, \exists y \in \mathbb{R}$telsque$x < y$ et $f(x) \leq f(y)$".

Correction de l'exercice 8 ▲

Remarquons d'abord que pour $n \in \mathbb{N}$, $\frac{2n+1}{n+2} \leq 2$ car $2n+1 \leq 2(n+2)$. Étant donné $\varepsilon > 0$, nous avons donc

$$\forall n \in \mathbb{N} \quad \frac{2n+1}{n+2} < 2 + \varepsilon$$

Maintenant nous cherchons une condition sur n pour que l'inégalité

$$2 - \varepsilon < \frac{2n+1}{n+2}$$

soit vraie.

$$\begin{aligned} 2-\varepsilon < \frac{2n+1}{n+2} &\Leftrightarrow (2-\varepsilon)(n+2) < 2n+1 \\ &\Leftrightarrow 3 < \varepsilon(n+2) \\ &\Leftrightarrow n > \frac{3}{\varepsilon} - 2 \end{aligned}$$

Ici ε nous est donné, nous prenons un $N \in \mathbb{N}$ tel que $N > \frac{3}{\varepsilon} - 2$, alors pour tout $n \geq N$ nous avons $n \geq N > \frac{3}{\varepsilon} - 2$ et par conséquent : $2 - \varepsilon < \frac{2n+1}{n+2}$. Conclusion : étant donné $\varepsilon > 0$, nous avons trouvé un $N \in \mathbb{N}$ tel que pour tout $n \geq N$ on ait $2 - \varepsilon < \frac{2n+1}{n+2}$ et $\frac{2n+1}{n+2} < 2 + \varepsilon$.
En fait nous venons de prouver que la limite de la suite de terme $(2n+1)/(n+2)$ tend vers 2 quand n tend vers $+\infty$.

Correction de l'exercice 9 ▲

$$\begin{aligned} x \in \complement(A \cup B) &\Leftrightarrow x \notin A \cup B \\ &\Leftrightarrow x \notin A \text{ et } x \notin B \\ &\Leftrightarrow x \in \complement A \text{ et } x \in \complement B \\ &\Leftrightarrow x \in \complement A \cap \complement B. \end{aligned}$$

$$\begin{aligned} x \in \complement(A \cap B) &\Leftrightarrow x \notin A \cap B \\ &\Leftrightarrow x \notin A \text{ ou } x \notin B \\ &\Leftrightarrow x \in \complement A \text{ ou } x \in \complement \\ &\Leftrightarrow x \in \complement A \cup \complement B. \end{aligned}$$

Correction de l'exercice 10 ▲

Nous allons démontrer l'assertion 1. de deux manières différentes.

1. Tout d'abord de façon "directe". Nous supposons que A et B sont tels que $A \cap B = A \cup B$. Nous devons montrer que $A = B$.

   Pour cela étant donné $x \in A$ montrons qu'il est aussi dans B. Comme $x \in A$ alors $x \in A \cup B$ donc $x \in A \cap B$ (car $A \cup B = A \cap B$). Ainsi $x \in B$.

   Maintenant nous prenons $x \in B$ et le même raisonnement implique $x \in A$. Donc tout élément de A est dans B et tout élément de B est dans A. Cela veut dire $A = B$.

2. Ensuite, comme demandé, nous le montrons par contraposition. Nous supposons que $A \neq B$ et non devons montrer que $A \cap B \neq A \cup B$.

   Si $A \neq B$ cela veut dire qu'il existe un élément $x \in A \setminus B$ ou alors un élément $x \in B \setminus A$. Quitte à échanger A et B, nous supposons qu'il existe $x \in A \setminus B$. Alors $x \in A \cup B$ mais $x \notin A \cap B$. Donc $A \cap B \neq A \cup B$.

Correction de l'exercice 11 ▲

Montrons quelques assertions.
$f(A \cap B) \subset f(A) \cap f(B)$.
Si $y \in f(A \cap B)$, il existe $x \in A \cap B$ tel que $y = f(x)$, or $x \in A$ donc $y = f(x) \in f(A)$ et de même $x \in B$ donc $y \in f(B)$. D'où $y \in f(A) \cap f(B)$. Tout élément de $f(A \cap B)$ est un élément de $f(A) \cap f(B)$ donc $f(A \cap B) \subset f(A) \cap f(B)$.

Remarque : l'inclusion réciproque est fausse. Exercice : trouver un contre-exemple.

$f^{-1}(F \setminus A) = E \setminus f^{-1}(A)$.

$$\begin{aligned} x \in f^{-1}(F \setminus A) &\Leftrightarrow f(x) \in F \setminus A \\ &\Leftrightarrow f(x) \notin A \\ &\Leftrightarrow x \notin f^{-1}(A) \quad \text{car } f^{-1}(A) = \{x \in E \ / \ f(x) \in A\} \\ &\Leftrightarrow x \in E \setminus f^{-1}(A) \end{aligned}$$

Correction de l'exercice 12 ▲

$I = [0,2]$ et $J =]1,+\infty[$.

Correction de l'exercice 13 ▲

Par l'absurde, supposons qu'il existe $p \in \mathbb{N}$ tel que $f = f_p$. Deux applications sont égales si et seulement si elles prennent les mêmes valeurs.

$$\forall n \in \mathbb{N} \ \ f(n) = f_p(n).$$

En particulier pour $n = p$, $f(p) = f_p(p)$. D'autre part la définition de f nous donne $f(p) = f_p(p) + 1$. Nous obtenons une contradiction car $f(p)$ ne peut prendre deux valeurs distinctes. En conclusion, quelque soit $p \in \mathbb{N}$, $f \neq f_p$.

Correction de l'exercice 14 ▲

1. Montrons en fait la contraposée.
   S'il existe i tel que p_i divise $N = p_1 p_2 \dots p_r + 1$ (i est fixé) alors il existe $k \in \mathbb{Z}$ tel que $N = k p_i$ donc

   $$p_i(k - p_1 p_2 \dots p_{i-1} p_{i+1} \dots p_r) = 1$$

   soit $p_i q = 1$ (avec $q = k - p_1 p_2 \dots p_{i-1} p_{i+1} \dots p_r$ un nombre entier). Donc $p_i \in \mathbb{Z}$ et $1/p_i = q \in \mathbb{Z}$, alors p_i vaut 1 ou -1. Et donc p_i n'est pas un nombre premier.
   Conclusion : par contraposition il est vrai que N n'est divisible par aucun des p_i
2. Raisonnons par l'absurde : s'il n'existe qu'un nombre fini r de nombres premiers $p_1, \dots, p_r$ alors $N = p_1 p_2 \dots p_r + 1$ est un nombre premier car divisible par aucun nombre premier autre que lui même (c'est le 1.).
   Mais N est strictement supérieur à tous les p_i. Conclusion on a construit un nombre premier N différent des p_i, il y a donc au moins $r+1$ nombres premiers, ce qui est absurde.

Correction de l'exercice 15 ▲

Rédigeons la deuxième égalité. Soit $\mathscr{A}_n$, $n \in \mathbb{N}^*$ l'assertion suivante :

$$(\mathscr{A}_n) \quad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$

– $\mathscr{A}_0$ est vraie ($1 = 1$).

– Étant donné $n \in \mathbb{N}^*$ supposons que $\mathscr{A}_n$ soit vraie. Alors

$$\begin{aligned}\sum_{k=1}^{n+1} k^2 &= \sum_{k=1}^{n} k^2 \ + (n+1)^2 \\ &= \frac{n(n+1)(2n+1)}{6} + (n+1)^2 \\ &= \frac{n(n+1)(2n+1)+6(n+1)^2}{6} \\ &= \frac{(n+1)(n(2n+1)+6(n+1))}{6} \\ &= \frac{(n+1)(n+2)(2(n+1)+1)}{6}\end{aligned}$$

Ce qui prouve $\mathscr{A}_{n+1}$.
– Par le principe de récurrence nous venons de montrer que $\mathscr{A}_n$ est vraie pour tout $n \in \mathbb{N}^*$.

Correction de l'exercice 16 ▲

1. Montrons la proposition demandée par récurrence : soit $\mathscr{A}_n$ l'assertion $f^{n+1} = f \circ f^n$. Cette assertion est vraie pour $n = 0$. Pour $n \in \mathbb{N}$ supposons $\mathscr{A}_n$ vraie. Alors

$$f^{n+2} = f^{n+1} \circ f = (f \circ f^n) \circ f = f \circ (f^n \circ f) = f \circ f^{n+1}.$$

Nous avons utiliser la definition de f^{n+2}, puis la proposition $\mathscr{A}_n$, puis l'associativité de la composition, puis la définition de f^{n+1}. Donc $\mathscr{A}_{n+1}$ est vraie. Par le principe de récurrence

$$\forall \in \mathbb{N} \ \ f^n \circ f = f \circ f^n.$$

2. On procède de même par récurrence : soit $\mathscr{A}_n$ l'assertion $(f^{-1})^n = (f^n)^{-1}$. Cette assertion est vraie pour $n = 0$. Pour $n \in \mathbb{N}$ supposons $\mathscr{A}_n$ vraie. Alors

$$(f^{-1})^{n+1} = (f^{-1})^n \circ f^{-1} = (f^n)^{-1} \circ f^{-1} = (f \circ f^n)^{-1} = (f^n \circ f)^{-1} = (f^{n+1})^{-1}.$$

Donc $\mathscr{A}_{n+1}$ est vraie. Par le principe de récurrence

$$\forall \in \mathbb{N} \ \ (f^{-1})^n = (f^n)^{-1}.$$

Correction de l'exercice 17 ▲

1. Montrons par récurrence $\forall n \in \mathbb{N}\ x_n > 3$. Soit l'hypothèse de récurrence :

$$(\mathscr{H}_n) : \quad x_n > 3.$$

- La proposition $\mathscr{H}_0$ est vraie car $x_0 = 4 > 3$.
- Soit $n \geq 0$, supposons $\mathscr{H}_n$ vraie et montrons que $\mathscr{H}_{n+1}$ est alors vraie.

$$x_{n+1} - 3 = \frac{2{x_n}^2 - 3}{x_n + 2} - 3 = \frac{2{x_n}^2 - 3x_n - 9}{x_n + 2}.$$

Par hypothèse de récurrence $x_n > 3$, donc $x_n + 2 > 0$ et $2{x_n}^2 - 3x_n - 9 > 0$ (ceci par étude de la fonction $x \mapsto 2x^2 - 3x - 9$ pour $x > 3$). Donc $x_{n+1} - 3$ et $\mathscr{H}_{n+1}$ est vraie.
- Nous avons montrer

$$\forall n \in \mathbb{N} \quad \mathscr{H}_n \Rightarrow \mathscr{H}_{n+1}$$

et comme $\mathscr{H}_0$ est vraie alors $\mathscr{H}_n$ est vraie quelque soit n. Ce qui termine la démonstration.

2. Montrons que $x_{n+1} - 3 - \frac{3}{2}(x_n - 3)$ est positif.

$$x_{n+1} - 3 - \frac{3}{2}(x_n - 3) = \frac{2{x_n}^2 - 3}{x_n + 2} - \frac{3}{2}(x_n - 3) = \frac{1}{2}\frac{{x_n}^2 + 3x_n + 12}{x_n + 2}$$

   Ce dernier terme est positif car $x_n > 3$.

3. Montrons par récurrence $\forall n \in \mathbb{N}\ x_n > \left(\frac{3}{2}\right)^n + 3$. Soit notre nouvelle l'hypothèse de récurrence :

$$(\mathscr{H}_n) \quad x_n > \left(\frac{3}{2}\right)^n + 3.$$

   - La proposition $\mathscr{H}_0$ est vraie.
   - Soit $n \geq 0$, supposons que $\mathscr{H}_n$ vraie et montrons que $\mathscr{H}_{n+1}$ est vérifiée.
     D'après la question précédente $x_{n+1} - 3 > \frac{3}{2}(x_n - 3)$ et par hypothèse de récurrence $x_n > \left(\frac{3}{2}\right)^n + 3$; en réunissant ces deux inégalités nous avons $x_{n+1} - 3 > \frac{3}{2}(\left(\frac{3}{2}\right)^n) = \left(\frac{3}{2}\right)^{n+1}$.
   - Nous concluons en résumant la situation :
     $\mathscr{H}_0$ est vraie, et $\mathscr{H}_n \Rightarrow \mathscr{H}_{n+1}$ quelque soit n. Donc $\mathscr{H}_n$ est toujours vraie.

4. La suite (x_n) tend vers $+\infty$ et n'est donc pas convergente.
