Calcul formel

20 février 2023

Table des matières

Chapitre 1

Introduction aux algorithmes

1.1 Introduction

D'après Wikipedia, « le calcul formel, ou parfois calcul symbolique, est le domaine des mathématiques et de l'informatique qui s'intéresse aux algorithmes opérant sur des objets de nature mathématique par le biais de représentations finies et exactes. Ainsi, un nombre entier est représenté de manière finie et exacte par la suite des chiffres de son écriture en base 2.

Étant données les représentations de deux nombres entiers, le calcul formel se pose par exemple la question de calculer celle de leur produit. Le calcul formel est en général considéré comme un domaine distinct du calcul scientifique, cette dernière appellation faisant référence au calcul numérique approché à l'aide de nombres en virgule flottante, là où le calcul formel met l'accent sur les calculs exacts sur des expressions pouvant contenir des variables ou des nombres en précision arbitraire. »

Dans ce cours, nous étudierons principalement les anneaux $\mathbb{Z}$ et $\mathbb{Z}/n\mathbb{Z}$, pour $n \in \mathbb{N}^*$. Nous nous intéresserons à différentes notions arithmétiques, comme le PGCD, les nombres premiers, le théorème des restes chinois ... et nous verrons quelques applications en cryptographie. Nous étudierons ces notions d'un point de vue algorithmique, c'est à dire que pour la plupart des quantités que nous introduirons, nous verrons des algorithmes « efficaces » qui permettent de les déterminer en pratique.

Définition 1.1. *Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une classe de problèmes.*

Exemples : Algorithmes vus à l'école primaire : addition de deux nombres entiers, multiplication de deux nombres entiers en la posant, division euclidienne, algorithme d'Euclide ...

Le terme *algorithme* provient du nom du mathématicien persan Al-Khwârismî, né dans les années 780 et mort vers 850. Cependant, les algorithmes sont antérieurs à ce mathématicien, puisque, par exemple, on a retrouvé une tablette d'argile datant d'entre 2000 et 1650 avant J.-C indiquant que les babyloniens avaient des méthodes algorithmiques pour calculer l'inverse de certains nombres (voir [Cha94]).

Dans ce chapitre, nous voyons quelques bases de programmation. Les exemples que nous donnerons seront rédigés en *pseudo-langage*, c'est à dire en langage « presque naturel », sans faire référence à un langage de programmation particulier. Lorsque nous implémenterons des algorithmes sur un ordinateur en revanche, nous utiliserons principalement *Sage*, qui est basé sur le langage *Python*.

1.2 Les types de données

1.2.1 Variables et valeurs

Pour mener à bien des calculs, il est nécessaire de manipuler des quantités données. Par exemple, pour calculer $P(2)$ avec $P(x) = x^2 - 1$, on remplace la valeur 2 partout où apparaît la variable x. Le vocabulaire de l'algorithmique garde les mêmes mots.

Une valeur est une quantité connue au moment du calcul. Le calcul concret va faire intervenir des valeurs. Par exemple, nous verrons souvent apparaître :
- des valeurs entières : 0, 72, -3, 25643, ...
- des valeurs flottantes (qui représentent des nombres réels) : 3.14159, 2.5, -21.876543, ...
- des valeurs booléennes : vrai ou faux
- des chaînes de caractères, par exemple : 'bonjour'

Une variable est un nom, qui va servir à décrire des calculs. En effet, en mathématiques on utilise la notation $P(x) = x^2 - 1$, et si on calcule l'image de x par la fonction $x \mapsto P(x)$, alors on devra porter x au carré et soustraire 1. Ceci reste vrai pour n'importe quelle valeur[1] $x = 1$, $x = \pi$ ou $x = -2.5$. En algorithmique, on utilisera aussi ce concept, et les variables serviront à stocker des valeurs.

Ceci permettra d'organiser le calcul par étapes, en stockant les résultats intermédiaires. On pourra imaginer qu'une variable est le nom d'un tiroir dans lequel on range une valeur, et dont on peut lire la valeur qui y est rangée.

Bien que $\mathbb{Z}$ soit inclus dans $\mathbb{R}$, il est souvent important de voir les entiers comme des entiers en Python et non comme des réels (cela se fait automatiquement en général). Cela permet par exemple de faire certaines opérations sur ces nombres, qui ne sont pas autorisées pour les réels (par exemple réduire un nombre modulo 2). Cela permet aussi de faire des calculs exacts et d'éviter les problèmes d'arrondis propres aux flottants. Par exemple pour Sage, par défaut, 1 est un entier, 10^{-15} est un nombre rationnel et $10^{-15} + 0.0$ est un réel ou flottant. Les calculs utilisant des nombres entiers ou rationnels sont exacts, mais ceux utilisant des nombres réels sont des approximations. Par exemple si on calcule $a = 1 + 0.0 + 10^{-15}$ en Sage, on obtient $a = 1.0$. Et si on calcule $a - 1$, Sage affiche $1.11022302462516e - 15$. En fait, dés que l'on rentre un nombre avec une virgule, Sage l'interprété comme un flottant, alors qu'un quotient de deux entiers est un rationnel.

1. Notons que ceci est vrai car le domaine de définition de $x \mapsto P(x)$ est $\mathbb{R}$ tout entier.

1.2.2 Les listes

Définition. *Une **liste** est une suite ordonnée d'expressions.*

Exemple. $[4, 5, 7]$, $[5, [2, 7]]$,...

La liste [ ] est appelée la **liste vide**.

Définition. *La **longueur** d'une liste est le nombre d'éléments de la liste.*

Exemples.
- *La longueur de* $[a_1, \ldots, a_n]$ *vaut* n.
- *La longueur de la liste vide* [ ] *vaut 0.*
- *La longueur de* $[3, 1, [2, 7], [3, 5, 6], [[2], 5]]$ *vaut 5.*

Généralement, les indices sont des entiers allant de 0 à la longueur de la liste à laquelle on retire 1.

On accède à une case du tableau en précisant son indice entre crochets.

Exemple. *Pour* $L = [0, 5, 7, 3, 12]$, , $L[0] = 0$ *et* $L[4] = 12$.

Attention !!!. *Bien vérifier, lorsqu'on accède à une case de la liste, que celle-ci existe bien. Par exemple,* $L[5]$ *renverra « erreur ».*

1.3 Les actions fondamentales

Un algorithme est une suite d'actions. On souhaite en faire une description précise et sans ambiguïté. Dans ce but, nous convenons dans cette partie d'un pseudo-langage en même temps que nous dressons la liste des actions fondamentales.

1.3.1 L'affectation

Plus haut, nous avons parlé de valeurs et de variables. L'affectation consiste à stocker une valeur dans une variable, et on la note

$$variable \leftarrow valeur.$$

Par exemple :

$$\begin{aligned} xEntier &\leftarrow 3 \\ xFlottant &\leftarrow 3.14159 \\ xBooleen &\leftarrow vrai \end{aligned}$$

Utilisation de la valeur stockée dans une variable Une fois qu'une variable a reçu une valeur, on peut utiliser la variable pour rappeler la valeur. Par exemple, si x contient la valeur 3, alors on peut former l'expression :

$$x + 3$$

qui est une valeur, et qui vaut 6.

Modification de la valeur stockée Il est important de comprendre que seule la variable à gauche de la flèche est modifiée. Ainsi, l'affectation :

$$x \leftarrow pam + 3$$

ne change que le contenu de la variable x, et pas celui de la variable pam. En effet, à droite de la flèche, on n'utilise pam que pour sa valeur. On peut aussi écrire :

$$\begin{aligned} x &\leftarrow 3 \\ x &\leftarrow x + 4 \end{aligned}$$

A la fin de ces deux instructions, la variable x contient la valeur 7.

Ordre des instructions Dans nos algorithmes, on pourra en général suivre les valeurs prises par les variables. Il faut toutefois garder à l'esprit que l'ordre dans lequel on effectue les instructions est important.

Par exemple, à la fin des instructions :

$$\begin{aligned} i &\leftarrow 12 \\ i &\leftarrow 34 \end{aligned}$$

la variable i contient la valeur 34.

A la fin des instructions :

$$\begin{aligned} i &\leftarrow 34 \\ i &\leftarrow 12 \end{aligned}$$

la variable i contient la valeur 12. Examinons un autre exemple. A la fin des instructions :

$$\begin{aligned} i &\leftarrow 34 \\ j &\leftarrow i + 2 \\ i &\leftarrow i + 1 \end{aligned}$$

la variable i contient la valeur 35 et la variable j contient la valeur 36. Par contre, à la fin des instructions :

$$\begin{aligned} i &\leftarrow 34 \\ i &\leftarrow i + 1 \\ j &\leftarrow i + 2 \end{aligned}$$

la variable i contient la valeur 35 et la variable j contient la valeur 37.

1.3.2 Les structures de sélection

Ces structures servent lorsqu'on est confronté à des situations pour lesquelles une ou plusieurs instructions ne peuvent être exécutées que dans certaines conditions.

L'instruction conditionnelle sert lorsqu'un bloc d'instructions ne peut être exécuté que si une condition est vraie.

Schéma. Condition(paramètres)
si condition
| faire...

Exemple : Algorithme de calcul (exact) des racines réelles d'un polynôme de degré 2 à coefficient dans $\mathbb{R}$. L'algorithme suivant prend en entrée trois réels (ou rationnels, ou entiers, ...) a, b, c, avec $a \neq 0$ et renvoie la liste des solutions de l'équation $ax^2 + bx + c = 0$ d'inconnue $x \in \mathbb{R}$.

Algorithme Racines(a, b, c)
Delta $\leftarrow b^2 - 4ac$
si Delta<0
| renvoyer [ ]
si Delta$=0$
| renvoyer $[-\frac{b}{2a}]$
si Delta>0
| renvoyer $[-\frac{b}{2a} + \sqrt{\frac{\text{Delta}}{2a}}, -\frac{b}{2a} - \sqrt{\frac{\text{Delta}}{2a}}]$.

Lorsqu'il y a un choix exclusif entre deux blocs d'instructions à exécuter, on peut également utiliser « sinon » (else)

Schéma. Alternatif(paramètres)
si condition
| faire...
sinon
| faire...

L'exemple suivant précise l'algorithme Pair.

Exemple. Pair(a)
si $a \bmod 2 = 0$
| renvoyer vrai
sinon
| renvoyer faux

Lorsqu'il y a un choix exclusif entre plusieurs blocs d'instructions, on peut aussi utiliser « sinon si » (elif dans Python).

Schéma. Multiple(paramètres)
si condition 1
  | faire...
sinon si condition 2
  | faire...
sinon si condition 3
  | faire...
...

1.3.3 Boucles itératives

1.3.3.1 Boucle « pour » (*for*)

Sert lorsqu'on doit répéter un nombre de fois déterminé la même tâche.

Schéma. For(paramètres)
pour i de (valeur de départ) à (valeur d'arrivée)
  | faire...

La variable i est appelée **compteur**.

A chaque passage dans la boucle, la valeur du compteur est automatiquement augmentée de 1.

Insistons sur le fait que nous nous **interdisons** de modifier la valeur de i par une affectation [2]. Si nous permettions de modifier i à l'intérieur de la boucle, il deviendrait très difficile de prédire le comportement de l'algorithme.

Par contre, dans les instructions, il est possible de faire intervenir la valeur de i. Lors du premier passage dans la boucle, i aura la valeur 1, puis au deuxième passage, elle aura la valeur 2, et ainsi de suite jusqu'à n.

Enfin, notons que :

— le nom du compteur de boucle importe peu, c'est-à -dire qu'il n'est pas obligatoire de l'appeler i ;

— le compteur de boucle peut prendre des valeurs qui sont dans des ensembles variés, entiers ou non, du moment qu'ils sont finis.

Exemple. Somme(n)
$s \leftarrow 0$
pour i de 1 à n
  | $s \leftarrow s + i$
renvoyer s

2. D'ailleurs, cette interdiction est mise en place dans les langages de programmation qui se soucient de la rigueur des programmes.

1.3.3.2 Boucle « tant que » (*while*)

Sert à exécuter une commande tant qu'une condition est vraie.

Attention !!!.

- *Avant la boucle, prévoir une condition initiale.*
- *Éviter les boucles infinies, et pour cela, vérifier que la boucle s'arrête à un moment de l'itération.*

Schéma.
While(paramètres)
condition initiale
tant que condition vraie
| faire...

Exemple.
Somme2(n)
$s \leftarrow 0$
$i \leftarrow 0$
tant que $i \leq n$
| $s \leftarrow s + i$
| $i \leftarrow i + 1$
renvoyer s

Exemple.
Division_euclidienne(n, b)
$r \leftarrow n$
$q \leftarrow 0$
tant que $r \geq b$
| $r \leftarrow r - b$
| $q \leftarrow q + 1$
renvoyer (q, r)

Exemple à éviter.
$n \leftarrow 0$
tant que $n \bmod 2 = 0$
| $n \leftarrow n + 2$
renvoyer n.

1.4 Récursivité

1.4.1 Définitions

Définition. *On appelle récursive toute fonction ou procédure qui s'appelle elle-même.*

Attention !!!. *Comme dans le cas d'une boucle, il faut un cas d'arrêt où l'on ne fait pas d'appel récursif.*

Schéma. Récursive(paramètres)
si cas particulier
| faire...
sinon
| ...
| Récursive(paramètres modifiés)
| ...

La fonction suivante décrit $n!$.

Exemple. Factorielle(n)
si $n = 0$
| renvoyer 1
sinon
| renvoyer $n\times$ Factorielle($n - 1$)

Un algorithme est dit **itératif** s'il n'est pas récursif.

Récursif ou itératif ? Un algorithme est dit itératif s'il n'est pas récursif. Lorsque l'on a un programme récursif, on peut souvent le réécrire en un programme récursif. Par exemple pour calculer $n!$, on peut aussi utiliser l'algorithme suivant.

Algorithme FactIt(n)
$a = 1$
$k \leftarrow 1$
tant que $k \leq n$:
| $a \leftarrow ak$
| $k \leftarrow k + 1$
Renvoyer a.

L'un des avantages des algorithmes récursifs est qu'ils sont parfois plus faciles à programmer, par exemple lorsque l'on utilise des récurrences. En revanche, ils utilisent souvent plus de mémoire que les algorithmes itératifs, ce qui peut amener à dépasser les capacités de l'ordinateur. Par exemple pour calculer 12! en récursif, l'ordinateur va devoir stocker 11!, 10!, ... jusqu'à 1. Alors qu'en itératif, il n'a besoin que de stocker a et k. De manière générale, pour un même problème, les algorithmes itératifs sont souvent plus efficaces que les algorithmes récursifs. Par exemple sous Python, le programme échoue à calculer (10 000)! en récursif, alors qu'il calcule sans problème (100 000)! en itératif.

Chapitre 2

Arithmétique dans $\mathbb{Z}$

2.1 Quelques références

— Combes, *Algèbre et géométrie*, [Com98],
— Cormen, Leiserson, Rivest Stein, *Introduction à l'algorithmique*, [CLRS02]
— Demazure, *Cours D'Algèbre*, [Dem08],
— Menezes, Katz, Van Oorschot, Vanstone, *Handbook of applied cryptographiy*, [MKVOV96],
— Saux Picart, *Cours de calcul formel : Algorithmes fondamentaux*, [Pic99].

2.2 Rappels sur l'anneau $\mathbb{Z}$

Rappels. *Soit $(A, +, .)$ un anneau commutatif.*

- *A est dit intègre si on a :*

$$\forall x, y \in A,\ xy = 0 \Leftrightarrow x = 0 \text{ ou } y = 0.$$

- *Soit $I \subset A$. I est appelé un idéal de A si :*
  - *$(I, +)$ est un sous-groupe de A.*
  - *$\forall (a, x) \in A \times I$, $ax \in I$.*
- *Un idéal I de A est dit premier si on a $x, y \in A$ tels que $xy \in I$, alors $x \in I$ ou $y \in I$.*

Définition 2.1. *Soit A un anneau intègre. On dit que A est **euclidien** si A est muni d'une application $v : A \to \mathbb{N} \cup \{-\infty\}$ telle que pour tous $(a, b) \in A \times (A \setminus \{0\})$, il existe $(q, r) \in A^2$ vérifiant $a = bq + r$ avec $v(r) < v(b)$.*

Remarque. *La raison d'être de $-\infty$ est de donner une valeur à $v(0)$.*

Théorème 2.2 (Division euclidienne dans $\mathbb{Z}$)**.** *L'anneau $\mathbb{Z}$ est euclidien, avec $v = |\,.\,|$. De plus, il y a existence et unicité du couple (q, r) tel que $r \in \mathbb{N}$.*

Remarque. *Le théorème précédent implique que :* $\forall (a,b) \in \mathbb{Z} \times \mathbb{Z}^*$, $\exists !(q,r) : a = bq + r$ *et* $0 \leq r < |b|$.

Le nombre q *est appelé le quotient de la division euclidienne de* a *par* b*, et* r *le reste, noté* $a \bmod b$.

Démonstration. Le fait que $\mathbb{Z}$ est intègre est déjà connu. Quitte à remplacer b par $-b$ et q par $-q$, on peut supposer que $b > 0$.

- **existence de** (q,r) Soit $X = \{a - bq \mid q \in \mathbb{Z}\} \cap \mathbb{N}$. Alors X est non-vide (car par exemple si $q = -ba^2$, $a - bq = a^2b^2 - a \in X$). Soit r le minimum de X. Alors $r \geq 0$ et $r < |b|$ (car sinon, $r - |b| \in X$ et $r - |b| < r$). Par définition, il existe $q \in \mathbb{Z}$ tel que $a = bq + r$.
- **unicité de** (q,r) : Supposons qu'on a $a = bq + r = bq' + r'$ tel que $0 \leq r < |b|$, $0 \leq r' < |b|$.
  On a alors $b(q - q') = r' - r$; or $-|b| < r' - r < |b|$, d'où $-|b| < b(q - q') < |b|$.
  On obtient $0 \leq |b||q - q'| < |b|$, i.e. $0 \leq |q - q'| < 1$.
  On obtient donc $q = q'$, et donc $r = r'$.

□

Remarques.

- *si* $b > 0$*, alors* $q = \lfloor a/b \rfloor$ *et* $r = a - bq$.
- *si* $b < 0$*, alors* $q = \lfloor a/b \rfloor + 1$ *et* $r = a - bq$.

La plupart des résultats qui suivent restent vrai dans n'importe quel anneau euclidien, par exemple $\mathbb{Q}[X]$ (en prenant v le degré du polynôme).

Théorème 2.3.

1) *(démonstration à connaître) Les idéaux de* $\mathbb{Z}$ *sont les parties de la forme* $n\mathbb{Z}$ *où* $n \in \mathbb{N}$.
2) $n\mathbb{Z} \subset m\mathbb{Z} \Leftrightarrow m \mid n$.
3) $n\mathbb{Z} = m\mathbb{Z} \Leftrightarrow m = n$*, pour* $m, n \in \mathbb{N}$.

Démonstration.

1) - $n\mathbb{Z}$ est un idéal de $\mathbb{Z}$: exercice.
     $(n\mathbb{Z}, +)$ est un sous-groupe de $(\mathbb{Z}, +)$ car $0 \in n\mathbb{Z}$ et pour $a, b \in n\mathbb{Z}$, il existe $a', b' \in \mathbb{Z}$ tels que $a = na'$, $b = nb'$; d'où $a - b = n(b - b') \in n\mathbb{Z}$ De plus, pour $x \in \mathbb{Z}$, $ax = na'x \in n\mathbb{Z}$, ce qui prouve que $n\mathbb{Z}$ est un idéal de $\mathbb{Z}$.
   - les idéaux de $\mathbb{Z}$ sont les $n\mathbb{Z}$.
     Soit $I \neq \emptyset$ un idéal non nul de $\mathbb{Z}$ et $x \in I \setminus \{0\}$. Si $x < 0$, alors $-x \in I \cap \mathbb{N}^*$ donc $I \cap \mathbb{N}^*$ est non vide. Soit $n = \min(I \cap \mathbb{N}^*)$.
     Par définition, comme $n \in I$, alors $n\mathbb{Z} \subset I$.
     Soient $a \in I$ et $r = a \mod n$. Alors $a = nq + r$ et $0 \leq r < n$. Donc $r = \underbrace{a}_{\in I} - \underbrace{nq}_{\in n\mathbb{Z} \subset I} \in I$, d'où $r \in I \cap \mathbb{N}$.
     Comme $r < n$, on en déduit que $r = 0$ et donc que $a = nq$. D'où $I = n\mathbb{Z}$.

2) " $\Rightarrow$ " $n\mathbb{Z} \subset m\mathbb{Z}$, donc $n \in m\mathbb{Z}$, i.e. $\exists k \in \mathbb{Z} : n = mk$, d'où $m \mid n$.
" $\Leftarrow$ " Soit $a \in n\mathbb{Z}$. Alors $\exists k \in \mathbb{Z} : a = nk$. De plus, $m \mid n$ par hypothèse donc $\exists k' \in \mathbb{Z} : n = mk'$. D'où $a = m(kk') \in m\mathbb{Z}$ et donc $a \in m\mathbb{Z}$.
3) Découle du 2).

□

Remarques.

- $a \in n\mathbb{Z} \iff n \mid a$.
- *Le* 1) *implique que* $\mathbb{Z}$ *est un anneau principal (i.e. que tout idéal est engendré par un seul élément).*
- *La relation* $\mid$ *est transitive, i.e si* $a, b, c \in \mathbb{Z}$ *sont tels quel* $a \mid b$ *et* $b \mid c$*, alors* $a \mid c$.

2.3 PGCD

2.3.1 Définition et caractérisation

Définition/Proposition 2.4. *Soient* $a, b \in \mathbb{Z}^*$. *Il existe un unique entier positif* d *tel que :*

1. d *divise* a *et* b,
2. *tout diviseur commun à* a *et* b *divise* d.

On appelle d *le plus grand commun diviseur (noté* $a \wedge b$*) de* a *et* b. *De plus,* d *est caractérisé par l'égalité*

$$\mathbb{Z}a + \mathbb{Z}b = d\mathbb{Z}.$$

En particulier, il existe $(u, v) \in \acute{}\mathbb{Z} \times \mathbb{Z}$ *tel que* $d = au + bv$ *(il s'agit de l'identité de Bézout).*

Démonstration. (à connaître)

- **Existence :** L'ensemble $a\mathbb{Z} + b\mathbb{Z}$ est un idéal de $\mathbb{Z}$ (exercice).
  Comme $\mathbb{Z}$ est principal (théorème 2.3), on en déduit l'existence de $d \in \mathbb{Z}$ tel que $d\mathbb{Z} = a\mathbb{Z} + b\mathbb{Z}$. Par définition, il existe $u, v \in \mathbb{Z}$ tels que $d = au + bv$. Quitte à remplacer d par $-d$, on peut supposer que $d \in \mathbb{N}$. Comme $\mathbb{Z}a \subset \mathbb{Z}a + \mathbb{Z}b = d\mathbb{Z}$, d divise a et par symétrie, d divise b. L'entier d est donc un diviseur commun à a et à b. Soit maintenant c un diviseur commun à a et à b. On a $\mathbb{Z}a \subset \mathbb{Z}c$ et $\mathbb{Z}b \subset \mathbb{Z}c$ et donc $\mathbb{Z}a + \mathbb{Z}b = \mathbb{Z}d \subset \mathbb{Z}c$, et donc c divise d.
- **Unicité :** Soit $d' \in \mathbb{N}$ vérifiant (1) et (2). Alors d' divise d et d divise d', donc $d = d'$.

□

Remarque. *(pour aller plus loin) L'existence du PGCD (ainsi que la plupart des résultats de ce chapitre) sont vrais dans n'importe quel anneau euclidien (par exemple* $\mathbb{R}[X]$ *ou* $\mathbb{Z}[i]$*). Par contre, un tel anneau* A *n'est en général pas muni d'un ordre. En fait dans la définition précédente, « grand » peut se définir « au sens de la division » : si* $a, b \in A$*, on dit que* a *est plus grand que* b *pour* $\mid$*, si* b *divise* a*. Alors la proposition précédente se généralise comme suit (exercice) :*

Pour tous $a, b \in A$, l'ensemble des diviseurs communs à a et b admet un plus grand élément d. On a $Aa + Ab = Ad$. Tout « plus grand diviseur commun à a et b » est de la forme xd, où x est un élément inversible de A.

2.3.2 Entiers premiers entre eux

Définition 2.5. *Soient $a, b \in \mathbb{Z}^*$. On dit que a et b sont **premiers entre eux** si $a \wedge b = 1$.*

Théorème 2.6 (Théorème de Bézout)**.** *Soient $a, b \in \mathbb{Z}^*$. Alors*

$$a \wedge b = 1 \iff \exists u, v \in \mathbb{Z} : au + bv = 1.$$

Démonstration. C'est une conséquence de la définition/proposition 2.4. □

Lemme 2.7. *(lemme de Gauss) Soient $a, b, c \in \mathbb{Z}^*$. Alors*

1) *Si $a \mid bc$ et $a \wedge b = 1$, alors $a \mid c$.*
2) *Si $a \wedge b = 1$, alors :*

$$a \mid c \text{ et } b \mid c \Rightarrow ab \mid c.$$

Démonstration. Exercice. □

2.4 Algorithme d'Euclide

2.4.1 Forme simple

Lemme 2.8. *Soient $a, b \in \mathbb{Z}^*$. Alors*

1. $a \wedge 0 = |a|$.
2. $a \wedge b = |a| \wedge |b|$.
3. $a \wedge b = (a - b) \wedge b$.
4. $a \wedge b = b \wedge r$ *si* $a = bq + r$ *avec* $(q, r) \in \mathbb{Z}^2$. *En particulier,* $a \wedge b = b \wedge (a \bmod b)$.

Démonstration. Les équations proviennent respectivement des égalités $a\mathbb{Z} + 0\mathbb{Z} = |a|\mathbb{Z}$, $a\mathbb{Z} = |a|\mathbb{Z}$, $a\mathbb{Z} + b\mathbb{Z} = (a - b)\mathbb{Z} + b\mathbb{Z}$ et $a\mathbb{Z} + b\mathbb{Z} = (a - bq)\mathbb{Z} + b\mathbb{Z} = r\mathbb{Z} + b\mathbb{Z}$. □

Théorème 2.9. *L'algorithme suivant permet de calculer le PGCD de 2 entiers $a > 0$ et $b \geq 0$.*

Algorithme 3. *Euclide(a, b)*
si $b = 0$
| *renvoyer a*
sinon
| *renvoyer Euclide(b,a mod b).*

Démonstration. Montrons que l'algorithme termine. On pose $a_0 = a$ et $b_0 = b$. Soit $i \in \mathbb{N}$ tel qu'on a défini a_i et b_i. Si $b_i = 0$, on pose $k = i$ et on s'arrête là . Sinon, on pose $a_{i+1} = b_i$ et $b_{i+1} = a_i \bmod b_i$. On a alors, $b_{i+1} < b_i$. On en déduit que tant que b_i est défini, on a $0 \leq b_i < b_{i-1} < \ldots < b_0 = b$. On a donc $i \leq b$ et donc l'algorithme termine (et le nombre $k+1$ d'étapes est inférieur ou égal à $b+1$).

De plus, si $i \in [\![1, k]\!]$, on a $a_i \wedge b_i = a_{i+1} \wedge b_{i+1}$ par le lemme 2.8. On en déduit que $a \wedge b = a_0 \wedge b_0 = a_k \wedge b_k = a_k \wedge 0 = a_k$. □

Exemple du fonctionnement de l'algorithme avec Euclide(27,15).

On montrera en TD que le nombre d'appels récursifs effectués par Euclide(a, b) est $O(\ln b)$.

2.4.2 Forme étendue de l'algorithme d'Euclide

Nous savons que si $d = PGCD(a, b)$, alors $\exists u, v \in \mathbb{Z} : d = au + bv$ $(*)$.

Question. *Peut-on trouver u et v ?*

Réponse. *Oui, grâce à une forme étendue de l'algorithme d'Euclide*

Soient $a, b \in \mathbb{N}$ et $d = a \wedge b$. Quitte à échanger a et b, on peut supposer que $b \leq a$. On veut trouver $u, v \in \mathbb{Z}$ tels que $au + bv = d$. Soient $r = a \bmod b$ et $q \in \mathbb{Z}$ tels que $a = bq + r$. Supposons que l'on connaisse $u', v' \in \mathbb{Z}$ tels que $bu' + rv' = d$. On a alors

$$d = bu' + rv' = bu' + (a - bq)v' = av' + b(u' - qv').$$

On en déduit donc u, v tels que $au + bv = d$. Comme dans le cas de l'algorithme d'Euclide, on en déduit que l'algorithme suivant, qui prend en entrée deux entiers $a > 0$ et $b \geq 0$ et renvoie un triplet d'entiers relatifs (d, u, v) vérifiant $d = a \wedge b$ et $au + bv = d$ est valide.

Algorithme Euclide-étendu(a, b)
si $b = 0$
  | renvoyer $(a, 1, 0)$
sinon
  | $(d', u', v') \leftarrow$ Euclide-étendu$(b, a \bmod b)$
  | $(d, u, v) \leftarrow (d', v', u' - \lfloor \frac{a}{b} \rfloor v')$
  | renvoyer (d, u, v)

L'exemple suivant sert à comprendre la procédure d'Euclide-étendu.

Exemple. *Étudions Euclide-étendu(105,78).*

a	b	$\lfloor \frac{a}{b} \rfloor$	d	u	v
105	78	1	3	3	$-1 - 1 \times 3 = -4$
78	27	2	3	-1	$1 - 2 \times (-1) = 3$
27	24	1	3	1	$0 - 1 \times 1 = -1$
24	3	8	3	0	$1 - 8 \times 0 = 1$
3	0		3	1	0

Nous pouvons trouver u et v également de la façon suivante :

$105 = 78 \times 1 + 27$ *donc* $3 = 27 - 24$
$78 = 27 \times 2 + 24$ $= 27 - (78 - 27 \times 2) = 27 \times 3 - 78$
$27 = 24 \times 1 + 3$ $= (105 - 78) \times 3 - 78 = 105 \times 3 - 78 \times 4$
$24 = 3 \times 8 + 0$

Les lignes 2 et 3 de l'algorithme déterminent l'arrêt du processus : si $b = 0$, alors $a = a \wedge b = a \times 1 + b \times 0$, et $(a, 1, 0)$ est alors renvoyé.

Si $b \neq 0$, Euclide-étendu calcule d'abord (d', u', v') tel que $d' = b \wedge (a \bmod b)$ et ainsi, $d' = bu' + (a \bmod b)v'$. On a alors $d = a \wedge b = b \wedge (a \bmod b) = d'$, et $d = d' = bu' + (a \bmod b)v' = bu' + (a - \lfloor \frac{a}{b} \rfloor b)v' = av' + b(u' - \lfloor \frac{a}{b} \rfloor v')$. Donc le choix $u = v'$ et $v = u' - \lfloor \frac{a}{b} \rfloor v'$ donne une solution à l'équation $d = au + bv$. L'algorithme est donc valable.

2.4.3 Équations diophantiennes linéaires

Nous avons ici une application directe de l'algorithme d'Euclide-étendu.

Proposition 2.10. *Soient $a, b \in \mathbb{Z}^*$ et $c \in \mathbb{Z}$. Considérons l'équation*

$$ax + by = c. \tag{2.1}$$

Alors (2.1) admet des solutions si et seulement si $d := a \wedge b$ divise c.

Dans ce cas, si $(u, v) \in \mathbb{Z}^2$ vérifie $au + bv = d$, alors $(x_0, y_0) = (uc/d, vc/d)$ est une solution de (2.1) et les solutions de (2.1) sont données par

$$\begin{cases} x = x_0 + \dfrac{kb}{d} \\ y = y_0 - \dfrac{ka}{d} \end{cases} \qquad (k \in \mathbb{Z}).$$

Démonstration. L'équation (2.1) admet une solution si et seulement si $c \in \mathbb{Z}a + \mathbb{Z}b = \mathbb{Z}d$, si et seulement si d divise c.

On suppose maintenant que d divise c. Soient $u, v \in \mathbb{Z}$ tels que $au + bv = d$. Alors $auc/d + bvc/d = dc/d = c$, donc (x_0, y_0) est bien solution de (2.1). Avec les notations de la proposition, on a $ax_0 + by_0 = (au + bv)\frac{c}{d} = c$, donc (x_0, y_0) est solution de (2.1).

Si (x, y) est solution de (2.1), alors $a(x - x_0) + b(y - y_0) = 0$. Si on pose $a' = \frac{a}{d}$ et $b' = \frac{b}{d}$, alors $a'(x - x_0) + b'(y - y_0) = 0$. Donc $b' \mid a'(x - x_0)$. Comme $a'u + b'v = 1$, on a $a' \wedge b' = 1$ donc d'après le lemme de Gauss, $b' \mid (x - x_0)$. D'où l'existence de $k \in \mathbb{Z}$ tel que $x - x_0 = kb'$.

On a $ka'b' + b'(y - y_0) = 0$, i.e. $y = y_0 - ka'$.

Conclusion : si (x, y) est solution de (2.1), alors :

$$\exists k \in \mathbb{Z} \mid (x, y) = (x_0 + kb', y_0 - ka').$$

Réciproquement, un tel couple est bien solution de (2.1). □

L'algorithme suivant donne, en prenant en entrée 3 entiers $(a, b, c) \in (\mathbb{N}^*)^2 \times \mathbb{N}$, une solution particulière de l'équation $ax + by = c$.

Algorithme 5. Diophantienne(a, b, c)
si c mod Euclide$(a, b) \neq 0$
| renvoyer « Pas de solution »
sinon
| $(d, u, v) \leftarrow$Euclide-étendu(a, b)
| renvoyer $(uc/d, vc/d)$.

2.5 Nombres premiers et décomposition en produit de facteurs premiers

Définition 2.11. *Soit $p \in \mathbb{Z} \setminus \{1, -1\}$. On dit que p est premier s'il a exactement quatre diviseurs.*

Comme tout nombre entier x admet $x, -x, 1$ et -1 comme diviseurs, un nombre p est premier si et seulement si $p \notin \{1, -1\}$ et si ses seuls diviseurs sont $1, -1, p, -p$.

Un nombre p est premier si et seulement si $-p$ est premier. On s'intéressera donc principalement aux nombres premiers positifs. On note $\mathbb{P}$ l'ensemble des nombres premiers positifs.

Lemme 2.12. *Soient $a \in \mathbb{Z}$ et $p \in \mathbb{P}$.*

1. *Supposons que p ne divise pas a. Alors $a \wedge p = 1$.*
2. *(lemme d'Euclide) Soit $b \in \mathbb{Z}$ tel que $p \mid ab$. Alors soit p divise a, soit p divise b.*

Démonstration. (1) Tout diviseur commun à a et à p divise p donc appartient à $\{1, -1, p, -p\}$.
(2) est une conséquence de (1) et du lemme de Gauss (lemme 2.7). □

Lemme 2.13. *Soit $n \in \mathbb{Z}$ tel que $|n| \geq 2$. Soit p son plus petit diviseur supérieur ou égal à 2. Alors p est premier. En particulier, n admet un diviseur premier.*

Démonstration. Soit q un diviseur de p différent de 1 ou -1. Alors $|q|$ divise p donc $|q|$ divise n donc $|q| \geq p$ donc $|q| = p$, ce qui prouve le lemme. □

Théorème 2.14. *(Euclide) Il existe une infinité de nombre premiers.*

Démonstration. Soit E un ensemble fini non vide de nombre premiers positifs et $a = (\prod_{p \in E} p) + 1$. Soient q un diviseur premier de a (on sait qu'il en existe par le lemme 2.13) et $p \in E$. On écrit $a = qa'$. Alors $qa' - p \prod_{p' \in E \setminus \{p\}} p' = 1$, donc p et q sont premiers entre eux. Ainsi, $q \notin E$ et donc $E \neq \mathbb{P}$. □

Théorème 2.15. *(Hadamard, La Vallée Poussin, 1896, admis) Pour $n \in \mathbb{N}$, on note $\pi(n)$ le nombre de nombres premiers compris entre 1 et n. Alors $\pi(n) \underset{n \to +\infty}{\sim} \ln(n)$.*

Théorème 2.16. *Soit $n \in \mathbb{Z}$ tel que $|n| \geq 2$. Alors il existe $\epsilon \in \{-1, 1\}$, $k \in \mathbb{N}$, $p_1 < \ldots < p_k \in \mathbb{P}$ des nombres premiers et $\alpha_1, \ldots, \alpha_k \in \mathbb{N}_{\geq 1}$ tels que $n = \epsilon p_1^{\alpha_1} \ldots p_k^{\alpha_k}$. De plus, cette décomposition est unique et $\{p_1, \ldots, p_k\}$ est l'ensemble des diviseurs premiers de n.*

Démonstration. Pour l'existence, on procède par récurrence sur $|n|$, en utilisant le fait que n admet un diviseur premier p et que $|\frac{n}{p}| < n$.

Démontrons l'unicité. Pour cela, on procède également par récurrence : on suppose qu'on connaît l'unicité de la décomposition pour tout $n' \in \mathbb{Z}$ tel que $|n'| < |n|$. On suppose que $n = \epsilon p_1^{\alpha_1} \ldots p_k^{\alpha_k} = \epsilon' q_1^{\beta_1} \ldots q_\ell^{\beta_\ell}$, avec $\epsilon, \epsilon' \in \{-1, 1\}$, $k, \ell \in \mathbb{N}$, $p_1 < p_2 < \ldots < p_k$, $q_1 < q_2 < \ldots < q_\ell$, $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_\ell \in \mathbb{N}^*$. Pour l'unicité, on procède également par récurrence en utilisant le lemme d'Euclide (lemme 2.12). Quitte à échanger les p et les q, on peut supposer que $q_1 \leq p_1$. Comme ϵ et ϵ' sont du signe de n, $\epsilon = \epsilon'$. Le nombre q_1 est premier et divise $p_1^{\alpha_1} \ldots p_k^{\alpha_k}$. Par le lemme d'Euclide (lemme 2.12), q_1 divise $p_1^{\alpha_1}$ ou q_1 divise $p_2^{\alpha_2} \ldots p_k^{\alpha_k}$. En réitérant le même raisonnement, on en déduit l'existence de $i \in [\![1, k]\!]$ tel que q_1 divise p_i. Comme p_i est premier, $q_1 = p_i$. On a alors $p_1 \leq p_i = q_1 \leq p_1$, donc $p_1 = q_1$. On a alors $p_1^{\alpha_1 - 1} p_2^{\alpha_2} \ldots p_k^{\alpha_k} = q_1^{\beta_1 - 1} q_2^{\beta_2} \ldots q_\ell^{\beta_\ell}$. Par l'unicité de la décomposition pour $n/\epsilon p$, on en déduit que $\alpha_1 - 1 = \beta_1 - 1$, $k = \ell$ et $\beta_j = \alpha_j$ pour $j \in [\![2, k]\!]$, ce qui démontre l'unicité de la décomposition pour n et achève la preuve du théorème. □

Corollaire 2.17. *Soit $n \in \mathbb{N}_{\geq 2}$ et $n = p_1^{\alpha_1} \ldots p_k^{\alpha_k}$ sa décomposition en produit de facteurs premiers (avec les même notations qu'au théorème 2.16). Alors l'ensemble des diviseurs de n est*

$$\{\epsilon p_1^{\alpha_1'} \ldots p_k^{\alpha_k'} \mid \epsilon \in \{-1, 1\}, \alpha_i' \in [\![0, \alpha_k]\!] \forall i \in [\![1, k]\!]\}.$$

Démonstration. Si $d = \epsilon p_1^{\alpha_1'} \ldots p_k^{\alpha_k'}$, avec $\alpha_i' \in [\![0, \alpha_i]\!]$, pour $i \in [\![1, k]\!]$, alors $d.\epsilon p_1^{\alpha_1 - \alpha_1'} \ldots p_k^{\alpha_k - \alpha_k'} = n$, donc d divise bien n. Réciproquement, soit d un diviseur de n. Soit q un diviseur premier de n. Alors par le lemme d'Euclide (lemme 2.12), $q \in \{p_1, \ldots, p_k\}$. On en déduit que d s'écrit $\epsilon p_1^{\beta_1} \ldots p_k^{\beta_k}$, où $\epsilon \in \{-1, 1\}$, $\beta_1, \ldots, \beta_k \in \mathbb{N}$. Soit $i \in [\![1, k]\!]$. Alors $p_i^{\beta_i}$ divise n. Supposons $\beta_i > \alpha_i$. Alors $p_i^{\beta_i - \alpha_i}$ divise $\prod_{j \neq i} p_j^{\alpha_j}$. C'est absurde par le lemme d'Euclide. Donc $\beta_i \leq \alpha_i$, d'où le résultat. □

Corollaire 2.18. *Soient $a, b \in \mathbb{N}_{\geq 2}$. On écrit $a = p_1^{\alpha_1} \ldots p_k^{\alpha_k}$ et $b = p_1^{\beta_1} \ldots p_k^{\beta_k}$, avec $p_1 < \ldots < p_k \in \mathbb{P}$ et $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k \in \mathbb{N}$. Alors $a \wedge b = p_1^{\gamma_1} \ldots p_k^{\gamma_k}$, avec $\gamma_i = \min\{\alpha_i, \beta_i\}$, pour tout $i \in [\![1, k]\!]$.*

Remarque.

- *La preuve du théorème 2.16 fournit un algorithme de décomposition d'un nombre en produit de facteurs premiers. En revanche, il est assez « long » (on verra plus tard ce que cela signifie). La difficulté dans cet algorithme est la recherche d'un facteur premier divisant n. Lorsque n est impair, la méthode la plus simple pour en trouver est de tester pour chaque nombre d entre 3 et $\sqrt{n}$ si d divise n, en avançant de 2 en 2, et en s'arrêtant lorsque l'on a trouvé un diviseur de n. On ne connaît d'algorithme « rapide » permettant la recherche d'un diviseur premier de n. Cette difficulté est utilisée en cryptographie.*

- *(pour aller plus loin) Le théorème 2.16 reste vrai dans n'importe quel anneau anneau A principal (sauf qu'en général on n'a plus d'ordre sur A et qu'il n'y a pas forcément de manière canonique pour ordonner $p_1, \dots, p_k$. Il y a donc unicité « à l'ordre des facteurs prés »). Les anneaux vérifiant cette propriété sont appelés anneaux factoriels et jouent un rôle important en arithmétique. Pour montrer que $\mathbb{Z}$ est factoriel, on a utilisé de manière cruciale la valeur absolue, qui permet de montrer le lemme 2.13 et de raisonner par récurrence. Pour montrer que tout anneau principal est factoriel, on peut montrer en préliminaire que dans un tel anneau, toute suite croissante d'idéaux $(I_k)_{k\in\mathbb{N}}$ est stationnaire (exercice). Voir [Com98, Chapitre 11], par exemple.*

pour aller plus loin)

Chapitre 3

Anneau $\mathbb{Z}/n\mathbb{Z}$

3.1 Généralités

Soit $n \in \mathbb{N}^*$. On munit $\mathbb{Z}$ de la relation d'équivalence $x \sim y$ si et seulement si $y-x \in n\mathbb{Z}$, pour $x, y \in \mathbb{Z}$. On note $\mathbb{Z}/n\mathbb{Z}$ l'ensemble quotient de $\mathbb{Z}$ par cette relation d'équivalence. Si $x \in \mathbb{Z}$, on note $\overline{x}$ sa classe (d'équivalence). On a alors $\overline{x} = x + n\mathbb{Z}$. De plus $\mathbb{Z}/n\mathbb{Z} = \{\overline{x} \mid x \in \mathbb{Z}\} = \{\overline{x} \mid x \in [\![0, n-1]\!]\}$ et cet ensemble a n éléments.

Définition/Proposition 3.1. *(exercice, cf TD) Soient $\overline{x}, \overline{y} \in \mathbb{Z}/n\mathbb{Z}$. On pose $\overline{x} + \overline{y} = \overline{x+y}$ et $\overline{x}.\overline{y} = \overline{xy}$. Ces opérations sont bien définies : si $x', y' \in \mathbb{Z}$ sont tels que $\overline{x} = \overline{x'}$ et $\overline{y} = \overline{y'}$, alors $\overline{x+y} = \overline{x'+y'}$ et $\overline{x}.\overline{y} = \overline{x'}.\overline{y'}$.*

Alors $(\mathbb{Z}/n\mathbb{Z}, +, .)$ est un anneau commutatif unitaire.

Proposition 3.2. *Soient $n \in \mathbb{N}^*$ et $x \in \mathbb{Z}$. Alors $\overline{x}$ est inversible dans $(\mathbb{Z}/n\mathbb{Z}, \times)$ si et seulement si $x \wedge n = 1$. Si $x \wedge n = 1$ et $u, v \in \mathbb{Z}$ sont tels que $xu + vn = 1$, alors $\overline{x}^{-1} = \overline{u}$.*

Démonstration. On a : $\overline{x} \in (\mathbb{Z}/n\mathbb{Z})^\times$ si et seulement si il existe $\overline{u} \in \mathbb{Z}/n\mathbb{Z}$ tel que $\overline{u}.\overline{x} = \overline{1}$ si et seulement si il existe $u, v \in \mathbb{Z}$ tels que $ux + vn = 1$ si et seulement si $x \wedge n = 1$. □

3.1.1 Sous-groupes de $(\mathbb{Z}/n\mathbb{Z}, +)$

Dans cette partie, on étudie les sous-groupes de $(\mathbb{Z}/n\mathbb{Z}, +)$.

Lemme 3.3. *Soit G un sous-groupe de $(\mathbb{Z}/n\mathbb{Z}, +)$. Alors G est monogène, c'est à dire qu'il existe $\overline{d} \in G$ tel que que $G = \langle \overline{d} \rangle = \mathbb{Z}\overline{d}$.*

Démonstration. (à connaître) Soit $I = \{a \in \mathbb{Z} \mid \overline{a} \in G\}$. Alors $0 \in I$ et si $a, b \in I$, $a - b = \overline{a} - \overline{b} \in G$. Donc $(I, 0)$ est un sous-groupe de $\mathbb{Z}$. De plus, si $k \in \mathbb{Z}$ et $a \in I$, $k\overline{a} = \overline{a} + \ldots + \overline{a} \in G$ (avec k termes dans l'addition), donc $ka \in I$. Donc I est un idéal de $\mathbb{Z}$, donc il existe $d \in \mathbb{Z}$ tel que $I = d\mathbb{Z}$ (car $\mathbb{Z}$ est principal, d'après le théorème 2.3). On a alors $\mathbb{Z}\overline{d} \subset G$ par définition de I et si $\overline{a} \in G$, $a \in \mathbb{Z}d$, donc $\overline{a} \in \mathbb{Z}\overline{d}$, et $G \subset \mathbb{Z}\overline{d}$. On a donc $G = \mathbb{Z}\overline{d}$. □

Lemme 3.4. *Soit $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$. Alors $\langle \overline{a} \rangle = \langle \overline{a \wedge n} \rangle$.*

Démonstration. Comme $\overline{a} = \frac{a}{a \wedge n} \overline{a \wedge n}$, on a $\langle \overline{a \wedge n} \rangle \supset \langle \overline{a} \rangle$. D'après le théorème de Bézout, il existe $u, v \in \mathbb{Z}$ tels que $au + nv = a \wedge n$. On a alors $\overline{au} = u\overline{a} = \overline{a \wedge n}$, donc $\langle \overline{a} \rangle \supset \langle \overline{a \wedge n} \rangle$, donc $\langle \overline{a} \rangle = \langle \overline{a \wedge n} \rangle$. □

Remarque 3.5. *Soit $a \in \mathbb{Z}$. Alors $\overline{a}$ engendre $\mathbb{Z}/n\mathbb{Z}$ (i.e $\langle \overline{a} \rangle = \mathbb{Z}/n\mathbb{Z}$) si et seulement si $a \wedge n = 1$. On retrouve le fait que $\overline{a}$ est inversible dans $\mathbb{Z}/n\mathbb{Z}$ si et seulement si $a \wedge n = 1$.*

exemple 3.6.
1. *Dans $\mathbb{Z}/15\mathbb{Z}$, $\langle \overline{6} \rangle = \{\overline{6}, \overline{12}, \overline{3}, \overline{9}, \overline{0}\} = \langle \overline{3} \rangle = \{\overline{0}, \overline{3}, \overline{6}, \overline{9}, \overline{12}\}$.*
2. *Dans $\mathbb{Z}/30\mathbb{Z}$, $\langle \overline{12} \rangle = \{\overline{0}, \overline{12}, \overline{24}, \overline{6}, \overline{18}\}$ et $\langle \overline{7} \rangle = \mathbb{Z}/30\mathbb{Z}$.*

Proposition 3.7.
1. *Les sous-groupes de $(\mathbb{Z}/n\mathbb{Z}, +)$ sont les $\langle \overline{d} \rangle$, où d est un diviseur de n.*
2. *Si d est un diviseur de n, alors $\langle \overline{d} \rangle = \{k\overline{d} \mid k \in [\![0, \frac{n}{d} - 1]\!]\}$, et cet ensemble est de cardinal $\frac{n}{d}$.*

Démonstration. 1) est une conséquence des lemmes 3.4 et 3.3.

2) Soit d un diviseur de n. Alors $\langle \overline{d} \rangle = \{k\overline{d} \mid k \in \mathbb{Z}\} \supset \{k\overline{d} \mid k \in [\![0, \frac{n}{d} - 1]\!]$. Soit $\overline{a} \in \langle d \rangle$. Écrivons $\overline{a} = \overline{kd}$, où $k \in \mathbb{Z}$. On effectue la division euclidienne de k par $\frac{n}{d}$: on a $k = q\frac{n}{d} + r$, où $q \in \mathbb{Z}$ et $r \in [\![0, \frac{n}{d} - 1]\!]$. Alors $\overline{kd} = \overline{qn + rd} = \overline{rd}$. On en déduit que $\langle \overline{d} \rangle = \{k\overline{d} \mid k \in [\![0, \frac{n}{d} - 1]\!]\}$. De plus, si $k, k' \in [\![0, \frac{n}{d} - 1]\!]$, $kd, k'd \in [\![0, \frac{n}{d} - 1]\!]$, donc $\overline{kd} = \overline{k'd}$ si et seulement si $k = k'$. □

Remarque 3.8. *On a $\langle \overline{d} \rangle = \{k\overline{d} \mid k \in \mathbb{Z}\} = \{k\overline{d} \mid k \in [\![-n, n]\!]\} = \ldots$ Il n'est donc pas clair a priori que le cardinal de l'ensemble de la proposition précédente est $\frac{n}{d}$.*

Lemme 3.9. *Soit $a \in \mathbb{Z} \setminus \{0\}$. Alors le sous-groupe $\{\overline{x} \in \mathbb{Z}/n\mathbb{Z} \mid \overline{ax} = \overline{0}\}$ est $\langle \overline{n/(a \wedge n)} \rangle$. On a aussi $\{\overline{x} \in \mathbb{Z}/n\mathbb{Z} \mid \overline{ax} = \overline{0}\} = \{\overline{x} \in \mathbb{Z}/n\mathbb{Z} \mid \overline{(a \wedge n)x} = \overline{0}\}$.*

Démonstration. Soit $x \in \mathbb{Z}$. Alors $\overline{ax} = \overline{0}$ si et seulement si $n \mid ax$ si et seulement si $\frac{n}{a \wedge n} \mid \frac{a}{a \wedge n} x$. Comme $\frac{a}{a \wedge n} \wedge \frac{n}{a \wedge n} = 1$, on peut appliquer le lemme de Gauss (lemme 2.7) : cette dernière condition est donc équivalente à $\frac{n}{a \wedge n} \mid x$, d'où le résultat. □

3.1.2 Équations linéaires modulaires

Dans cette partie, on fixe $n \in \mathbb{N}^*$. Soient $a, b \in \mathbb{N}^*$. Le but ici est de résoudre l'équation :

$$ax \equiv b[n], \tag{3.1}$$

d'inconnue $x \in \mathbb{Z}$. Soit $x \in \mathbb{Z}$. Alors x est solution de (3.1) si et seulement si $\overline{x}$ est solution de l'équation :

$$\overline{a}.\overline{x} = \overline{b}. \tag{3.2}$$

On va donc surtout étudier l'équation (3.2).

Remarque. *1.* (3.2) *admet des solutions si et seulement si* $\overline{b} \in \langle \overline{a} \rangle$ *où* $\langle \overline{a} \rangle$ *désigne le sous-groupe de* $\mathbb{Z}/n\mathbb{Z}$ *engendré par* $\overline{a}$*,*

2. Supposons que $\overline{a}$ *est inversible. Alors* $\overline{a}.\overline{a}^{-1} = \overline{1}$ *et* $\overline{a}.\overline{x} = \overline{b}$ *si et seulement si* $\overline{x} = \overline{b}.\overline{a}^{-1}$*, si* $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$*. Ainsi* (3.2) *admet une unique solution. On va s'intéresser au cas général, c'est à dire le cas où* $\overline{a}$ *n'est pas nécessairement inversible.*

Proposition 3.10. *Soient* $a, b, n \in \mathbb{N}^*$*. On pose* $d = a \wedge n$*.*

L'équation (3.2) *a des solutions si et seulement si* $d \mid b$*. Dans ce cas, si on choisit* x' *et* y' *tels que* $d = ax' + ny'$*, alors* (3.2) *admet exactement* d *solutions distinctes qui sont données par*

$$\overline{x_i} = \overline{x_0} + k\overline{n/d}, \quad (0 \leq k \leq d-1) \quad \text{où } x_0 = x'\frac{b}{d}.$$

Démonstration.

- (3.1) admet des solutions si et seulement si $d \mid b$.

$$\begin{aligned} \exists x \in \mathbb{Z} :\ ax \equiv b[n] &\iff \exists x, y \in \mathbb{Z} :\ ax + ny = b \\ &\iff b \in a\mathbb{Z} + n\mathbb{Z} = d\mathbb{Z} \iff d \mid b. \end{aligned}$$

- $\overline{x_0}$ solution de (3.1).
  Si $d = ax' + ny'$, alors :

$$ax_0 = a\frac{b}{d}x' = \frac{b}{d}(ax') = \frac{b}{d}(d - ny') = b - n\frac{b}{d}y' \equiv b[n],$$

  donc $\overline{x_0}$ est solution de (3.2).
- conclusion Soit $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$. Alors $\overline{x}$ est solution de (3.2) si et seulement si $\overline{a}.\overline{x} = \overline{b} = \overline{a}.\overline{x_0}$ si et seulement si il existe $k \in [\![0, d-1]\!]$ tel que $\overline{x} - \overline{x_0} = k\overline{n/d}$, d'après le lemme 3.9.

□

Exemple. *L'équation* $15x \equiv 6[21]$ *admet des solutions car* $15 \wedge 21 = 3 \mid 6$*.*

Après calcul, on a $3 = 15 \times 3 - 21 \times 2$*, donc les solutions de cette équation dans* $\mathbb{Z}/21\mathbb{Z}$ *sont* $x_0 = 3 \times \frac{6}{3} = 6$*,* $x_1 = 13$ *et* $x_2 = 20$*.*

3.2 Le théorème des restes chinois

Théorème 3.11 (Restes chinois)**.** *Soit* $n \in \mathbb{N}$*. On suppose que* n *s'écrit* $n = n_1 \dots n_k$ *où les* n_i *sont deux à deux premiers entre eux. Soit*

$$\begin{array}{rccc} \tilde{\phi} : & \mathbb{Z} & \longrightarrow & \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z} \\ & a & \longmapsto & (a \bmod n_1, \dots, a \bmod n_k) \end{array}$$

$: \mathbb{Z} \to \mathbb{Z}/n_1\mathbb{Z} \times \ldots \times \mathbb{Z}/n_k\mathbb{Z}$*. Alors* $\tilde{\phi}$ *induit un isomorphisme d'anneaux :*

$$\begin{array}{rccc} \phi : & \mathbb{Z}/n\mathbb{Z} & \longrightarrow & \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z} \\ & a \bmod n & \longmapsto & \tilde{\phi}(a) = (a \bmod n_1, \dots, a \bmod n_k) \end{array}$$

Démonstration.

- ϕ est bien défini :
- ϕ injectif.
  Soit $a \in \mathbb{Z}$ tel que $\phi(a \bmod n) = 0$. Alors n_i divise a pour tous $i \in [\![1, k]\!]$, donc par le lemme de Gauss (lemme 2.7), $n = n_1 \dots n_k$ divise a. D'où $a \bmod n = 0$, ce qui prouve que ϕ est injectif.
- ϕ surjectif.
  On a $|\mathbb{Z}/n\mathbb{Z}| = n = |\mathbb{Z}/n_1\mathbb{Z} \times \dots \times \mathbb{Z}/n_k\mathbb{Z}|$ et ϕ est injectif donc ϕ est nécessairement surjectif, ce qui prouve que c'est un isomorphisme. On donne ici une méthode plus constructive pour déterminer ϕ^{-1}.
  Pour $i \in [\![1, k]\!]$, on pose $\hat{n}_i = n/n_i = \prod_{j \neq i} n_j$. Alors $n_i \wedge \hat{n}_i = 1$, donc il existe $u_i, v_i \in \mathbb{Z}$ tels que $u_i \hat{n}_i + v_i n_i = 1$. Posons $c_i = u_i \hat{n}_i \bmod n$. Alors $\phi(c_i)$ est le vecteur dont les coordonnées sont nulles sauf la i-éme qui vaut 1 mod n_i. Soit $x = (x_1 \bmod n_1, \dots, x_k \bmod n_k) \in \mathbb{Z}/n_1\mathbb{Z} \times \dots \times \mathbb{Z}/n_k\mathbb{Z}$. Alors $\phi(\sum_{i=1}^{k} x_i c_i) = x$, ce qui prouve que ϕ est surjectif.

□

Remarque 3.12. *Dans le théorème précédent il est fondamental que les n_i soient deux à deux premiers entre eux. Par exemple $\mathbb{Z}/4\mathbb{Z}$ n'est pas isomorphe à $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. En effet,* 1 mod 4 *est d'ordre* 4 *dans $\mathbb{Z}/4\mathbb{Z}$ alors que tous les éléments de $\mathbb{Z}/2\mathbb{Z}$ sont d'ordres* 1 *ou* 2*.*

Exemple. *Soit $\phi : \mathbb{Z}/12\mathbb{Z} \to \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ l'isomorphisme chinois. On a $\phi(\overline{0}) = (\overline{0}, \overline{0})$, $\phi(\overline{1}) = (\overline{1}, \overline{1})$, $\phi(\overline{2}) = (\overline{2}, \overline{2})$, $\phi(\overline{3}) = (\overline{3}, \overline{3}) = (\overline{0}, \overline{3})$, $\phi(\overline{4}) = (\overline{4}, \overline{4}) = (\overline{1}, \overline{0})$, $\phi(\overline{5}) = (\overline{2}, \overline{1})$, $\phi(\overline{6}) = (\overline{0}, \overline{2})$, $\phi(\overline{7}) = (\overline{1}, \overline{3})$, $\phi(\overline{8}) = (\overline{2}, \overline{0})$, $\phi(\overline{9}) = (\overline{0}, \overline{1})$, $\phi(\overline{10}) = (\overline{2}, \overline{1})$, $\phi(\overline{11}) = (\overline{3}, \overline{2})$.*

Corollaire 3.13. *Soit $n = n_1 \dots n_k$ où les n_i sont deux à deux premiers entre eux. Alors pour tout $a_1, \dots, a_k \in \mathbb{Z}$, le système d'équations*

$$\begin{cases} x & \equiv & a_1[n_1] \\ & \vdots & \\ x & \equiv & a_k[n_k] \end{cases} \tag{3.3}$$

possède une unique solution modulo n.

Démonstration. Soit $x \in \mathbb{Z}$. Alors x est solution de (3.3) si et seulement si $\phi(x \bmod n) = (a_1, \dots, a_k)$. Alors en reprenant les notations du théorème 3.11, $\phi(x) = (a_1 \bmod n_1, \dots, a_k \bmod n_k)$. Comme ϕ est bijective, alors x existe bien et il est unique modulo n en tant qu'antécédent de $(a_1 \bmod n_1, \dots, a_k \bmod n_k)$ par ϕ. □

Corollaire 3.14. *Soient $n, n_1, \dots, n_k$ comme dans le théorème précédent. Alors pour $x, a \in \mathbb{Z}$, on a*

$$\big(x \equiv a[n_i] \quad (1 \leq i \leq k)\big) \Longleftrightarrow x \equiv a[n].$$

Systéme d'équations modulaires

Exemple. *Trouver les entiers x tels que $x \equiv 2[5]$ et $x \equiv 3[13]$.*

On a $2.13 - 5.5 = 1$ donc avec les notations de la preuve du théorème 3.11, on a $c_1 = 26 \bmod 65$ et $c_2 = -25 \bmod 65$. On a donc

$$\phi^{-1}(2 \bmod 5, 3 \bmod 13) = 2.c_1 + 3.c_2 = -23 \bmod 65 = 42 \bmod 65.$$

On en déduit que $\{x \in \mathbb{Z} \mid x \equiv 2[5] \text{ et } x \equiv 3[13]\} = 42 + 65\mathbb{Z}$.

Pour le cas général, c'est-à -dire pour la résolution d'un système (3.3), on peut utiliser un algorithme récursif. On résout d'abord pour les deux premières équations : on obtient un nombre α. On obtient

$$\begin{cases} x & \equiv & \alpha[n_1 n_2] \\ x & \equiv & a_3[n_3] \\ & \vdots & \\ x & \equiv & a_k[n_k] \end{cases}$$

et on continue récursivement pour obtenir un nombre dans $[0, n_1 \dots n_k]$.

3.3 Algorithme de Garner

Un inconvénient de l'algorithme d'inversion de l'isomorphisme du théorème des restes chinois (théorème 3.11) est le fait que pour $y \in \mathbb{Z}/n_1\mathbb{Z} \times \dots \mathbb{Z}/n_k\mathbb{Z}$ (avec les notations du théorème 3.11) on obtient comme antécédent un élément $\overline{x} \in \mathbb{Z}/n_1 \dots n_k\mathbb{Z}$, avec $x \in \mathbb{Z}$ pouvant être beaucoup plus grand que $n_1 \dots n_k$. Il faut ensuite faire une division euclidienne par $n_1 \dots n_k$ pour obtenir un antécédent dans $[\![0, n_1 \dots n_k - 1]\!]$.

Exemple. *On étudie le système*

$$\begin{cases} x & \equiv & 1000[1891] \\ x & \equiv & 2000[2499] \end{cases} \tag{3.4}$$

Après calcul, on obtient Euclide-étendu$(2499, 1891) = (1, 762, -1007) = (1, u, v)$, et on obtient alors comme solution particulière $x = 2000 \times 1891 \times (-1007) + 1000 \times 762 \times 2499 = -1904236000$, qui comporte dix chiffres, alors que $1891 \times 2499 = 4725609$ n'en a que sept.

Dans cette partie, on décrit l'algorithme de Garner, qui permet d'obtenir directement un antécédent dans $[\![0, n_1 \dots n_k - 1]\!]$.

3.3.1 Base mixte

Nous allons voir ici un autre système de représentants d'entiers, qui se révélera plus adapté à la résolution des systèmes d'équations modulaires.

Théorème 3.15. *Soient $n_1, \dots, n_k$ des entiers strictement positifs (non nécessairement distincts). On pose $n_0 = 1$. Alors la fonction*

$$\begin{array}{rccl} \psi : & [\![0, n_1[\![\times \dots \times [\![0, n_k[\![& \longrightarrow & \left[\!\left[0, \prod_{i=1}^{k} n_i\right[\!\right[\\ & (a_1, \dots, a_k) & \longmapsto & \displaystyle\sum_{i=1}^{k} a_i \prod_{j=0}^{i-1} n_j = a_1 + a_2 n_1 + a_3 n_1 n_2 + \dots \end{array}$$

est bien définie et est bijective.

Démonstration.

- ψ est bien définie.
  On raisonne par récurrence sur k.
  - $a_1 \in [\![0, n_1[\![$
  - soit $k' \in [\![1, k-1]\!]$ tel que pour tous $(a_1, \dots, a_{k'}) \in \prod_{i=1}^{k'} [\![0, n_i[\![$, on ait $\sum_{1 \le i \le k} a_i \prod_{0 \le j \le i-1} n_j \in [\![0, \prod_{1 \le i \le k'} n_j[\![$. Soit $a_{k'+1} \in [\![0, n_{k'+1}[\![$. Alors

$$\begin{aligned} \sum_{i=1}^{k'+1} a_i \prod_{j=0}^{i-1} n_j = \sum_{i=1}^{k'} a_i \prod_{j=0}^{i-1} n_j + a_{k'+1} \prod_{j=0}^{k'} n_j < \prod_{j=1}^{k'} n_j + a_{k'+1} \prod_{j=1}^{k'} n_j &= (a_{k'+1}+1) \prod_{j=1}^{k'} n_j \\ &\le \prod_{j=1}^{k'+1} n_j. \end{aligned}$$

  On a donc bien $\sum_{i=1}^{k'+1} a_i \prod_{j=0}^{i-1} n_j \in [\![0, \prod_{j=0}^{k'+1} n_j[\![$ et par récurrence on en déduit que ψ est bien définie.
- ψ est surjective.
  Soit $a \in [\![0, \prod_{1 \le i \le k} n_i[\![$. On pose $q_0 = a$. Alors plusieurs divisions euclidiennes nous donnent :

$$\begin{cases} q_0 = a &= q_1 n_1 + a_1 \\ q_1 &= q_2 n_2 + a_2 \\ &\vdots \\ q_{k-1} &= q_k n_k + a_k, \end{cases}$$

  avec $q_i \in \mathbb{N}$ et $a_i \in [\![0, n_i[\![$ pour tout $i \in [\![1, k]\!]$.
  Alors :

$$\begin{aligned} a &= a_1 + n_1(a_2 + n_2(\dots + n_{k-1}(a_k + q_k n_k)\dots)) \\ &= \sum_{i=1}^{k} a_i \prod_{j=0}^{i-1} n_j + q_k \prod_{i=1}^{k} n_i. \end{aligned}$$

  En particulier, comme $a \in [\![0, \prod_{1 \le i \le k} n_i[\![$, on a $q_k = 0$. Ainsi, $\psi(a_1, \dots, a_k) = a$.

⋆ Conclusion
Comme le cardinal de l'ensemble de départ de ψ est égal à celui de son ensemble d'arrivée, alors ψ est bijective.

□

Définition 3.16. *L'écriture d'un entier a vérifiant $0 \leq a < \prod_{1\leq i\leq k} n_i$ sous la forme $a = a_1 + a_2n_1 + a_3n_1n_2 + \ldots + a_kn_1 \ldots n_{k-1}$ s'appelle l'**écriture en base mixte** de a.*

Remarques.

- *Lorsque tous les n_i sont égaux à un certain $b \in \mathbb{N}_{\geq 2}$, on obtient le développement en base b usuel.*
- *Dans le cas où les n_i sont deux à deux premiers entre eux, on dispose de deux systèmes de numération des éléments $[\![0, \prod_{1\leq i\leq k} n_i[\![$: celle donnée par le théorème chinois et celle donnée par l'écriture en base mixte.*
- *Ces deux représentations sont distinctes et ont des propriétés distinctes. En effet, par exemple, lorsque $k = 3$, $n_1 = 2$, $n_2 = 5$, $n_3 = 7$, et qu'on choisit $a = 51(< 70)$, on a alors $\phi(a) = (1, 1, 2)$ tandis que $\psi^{-1}(a) = (1, 0, 5)$.*

3.3.2 L'algorithme

Commençons par donner le principe de l'algorithme de Garner. Pour cela, on part des données du système (3.3). Soit $x \in [\![0, n_1 \ldots n_k[\![$ une solution de (3.3). On code x dans la base mixte ; on sait qu'il existe des entiers $0 \leq \nu_i < n_i$ pour tout $1 \leq i \leq k$ tels que x s'écrive sous la forme

$$x = \sum_{i=1}^{k-1} \nu_i \prod_{j=1}^{i-1} n_j,$$

et on va déterminer les ν_i les uns après les autres.

On a d'abord $\nu_1 \equiv a_1[n_1]$.

Supposons à présent qu'on connaisse les ν_i pour $1 \leq i \leq j-1$. Comme $x \equiv a_j[n_j]$, on a alors

$$a_j \equiv \sum_{i=1}^{j} \nu_i \prod_{\ell=0}^{i-1} n_\ell \ [n_j].$$

Or $\prod_{1\leq i\leq j-1} n_i$ est premier avec n_j et est donc inversible modulo n_j. On obtient alors

$$\nu_j \bmod n_j = \Big((\prod_{i=1}^{j-1} n_i) \bmod n_j\Big)^{-1} \Big(a_j - \sum_{i=1}^{j-1} \nu_i \prod_{\ell=0}^{i-1} n_\ell\Big) \bmod n_j.$$

Cette démarche nous permet d'écrire à présent l'algorithme ci-dessous qui, prend en entrée deux listes $N = [n_1, \ldots, n_k]$, où les n_i sont deux à deux premiers entre eux et $A = [a_1, \ldots, a_k]$, et qui renvoie la liste $[\nu_1, \ldots, \nu_k]$ et le plus petit entier naturel x qui soit solution du système (3.3).

Algorithme 8. Garner(A, N)
$M \leftarrow [A[0]]$
$(x, p) \leftarrow (0, 1)$
$k \leftarrow long(N)$
pour i de 1 à $k-1$
  | $x \leftarrow x + M[i-1] * p$
  | $p \leftarrow p * N[i-1]$
  | $m \leftarrow \text{Inverse}(p, N[i]) * (A[i] - x) \bmod N[i]$
  | $M \leftarrow [int(M), m]$
$x \leftarrow x + M[k-1] * p$
renvoyer (M, x)

Exemple. *1. On veut résoudre le système* $\begin{cases} x \equiv 0[3] \\ x \equiv 3[5] \\ x \equiv 1[7] \end{cases}$ *d'inconnue* $x \in \mathbb{Z}$. *Comme* $3, 5$ *et* 7 *sont deux à deux premiers entre eux, il existe une unique solution* x *modulo* $3.5.7 = 105$. *On cherche donc une solution* $x \in [\![0, 104]\!]$. *D'après le théorème de la base mixte, on peut écrire* $x = \nu_1 + 7\nu_2 + 7.5\nu_3 = \nu_1 + 7\nu_2 + 35\nu_3$, *où* $\nu_1 \in [\![0, 6]\!]$, $\nu_2 \in [\![0, 4]\!]$ *et* $\nu_3 \in [\![0, 2]\!]$. *Déterminons* ν_1, ν_2 *et* ν_3.
On a $x = \nu_1 + 7\nu_2 + 35\nu_3 \equiv \nu_1 \equiv 1[7]$, *donc* $\nu_1 = 1$. *On a* $x = 1 + 7\nu_2 + 35\nu_3 \equiv 1 + 7\nu_2 \equiv 3[5]$, *donc* $2\nu_2 \equiv 2[5]$, *donc* $\nu_2 \equiv 1[5]$. *On a donc* $\nu_2 = 1$, *donc* $x = 1 + 7 + 35\nu_3 \equiv 0[3]$, *donc* $-\nu_3 \equiv -8 \equiv 1[3]$, *donc* $\nu_3 = 2$. *On en déduit que* $x = 1 + 7 + 2.35 = 78$. *L'ensemble des solutions du système est donc* $78 + 105\mathbb{Z}$.

2. Reprenons ici notre système (3.4).
Avec l'algorithme de Garner, on cherche x *sous la forme* $x = \nu_1 + 1891\nu_2$ *où* $0 \leq \nu_1 < 1891$ *et* $0 \leq \nu_2 < 2499$. *On a* $\nu_1 = 1000$ *et, après calcul,* $\nu_2 = (1891)^{-1}(2000 - 1000) \equiv 194[2499]$, *d'où* $x = 184427$.

Chapitre 4

Introduction à la complexité

4.1 Notions de complexité

4.1.1 Coût d'un algorithme

Cette introduction est fortement inspirée de [Pic99].

Pour un concepteur d'algorithme, une tâche capitale est de pouvoir a priori évaluer le temps que prendra son algorithme pour fournir le résultat escompté sur une machine donnée. Il faut en effet prendre conscience du fait que de nombreux algorithmes et formules (une formule n'étant qu'un algorithme particulier) n'ont qu'une valeur théorique et ne débouchent sur aucun calcul pratique car leur exécution demanderait beaucoup trop de temps. Par exemple, pour calculer le déterminant d'une matrice 25×25 à coefficients entier (qu'on peut même supposer entre 0 et 9) $A = (a_{i,j})_{1 \leq i,j \leq 25}$, on peut envisager d'utiliser la formule suivante :

$$\det(A) = \sum_{\sigma \in \mathfrak{S}_n} \epsilon(\sigma) a_{\sigma(1),1} \dots a_{\sigma(25),25},$$

où $\mathfrak{S}_{25}$ désigne le groupe des permutations de l'ensemble $[\![1, 25]\!]$. L'application de cette formule nécessite d'effectuer $|\mathfrak{S}_{25}| = 25! \simeq 1.55.10^{25}$ produits. Si l'on dispose d'un ordinateur capable d'effectuer 10 milliards de ces produits par seconde, il faudrait 49 millions d'années pour effectuer ce calcul. C'est donc inenvisageable en pratique, même en tenant compte des progrès techniques à venir (à court terme) des processeurs. En fait l'application directe de la définition pour le calcul du déterminant d'une matrice de taille n nécessite de faire $n!$ multiplications et additions, alors qu'une méthode relativement simple basée sur le pivot de Gauss permet de l'effectuer en $O(n^3)$ multiplications et additions. Il est donc important d'évaluer le temps que nécessitera un algorithme pour fournir le résultat avant même de le programmer. De plus, on aimerait pouvoir comparer autant que faire se peut, les algorithmes entre eux, et ceci de manière aussi intrinsèque que possible, c'est-à -dire, sans que notre mesure dépende de l'ordinateur ou du langage de programmation que l'on utilisera.

Nous supposons que nous sommes en possession d'un ordinateur dont le micro-processeur est capable de calculer directement la somme, la différence et le produit de toute paire de

nombres entiers naturels inférieurs à un certain entier b. Nous supposons aussi qu'il peut réaliser les divisions d'un nombre à deux chiffres par un nombre à un chiffre en base b, fournissant en une seule commande le quotient et le reste. Nous appelons *opérations élémentaires* ces quatre opérations définies sur $[\![0, b-1]\!]$, réalisables par le micro-processeur en une seule commande. En pratique, b dépend du processeur de l'ordinateur sur lequel on travaille. Actuellement, pour la plupart des ordinateurs grands public, $b = 2^{64}$. Pour nous, la valeur de b n'est pas très importante car on cherchera à évaluer les algorithmes à une constante multiplicative prés qui dépend du processeur utilisé.

On choisit une base $b \in \mathbb{N}_{\geq 2}$ dans laquelle on représente les entiers naturels. Tout $N \in \mathbb{N}^*$ s'écrit $N = \sum_{i=0}^{k} a_i b_i$, avec $a_i \in [\![0, b-1]\!]$ et $a_k \neq 0$. On a $b^k \leq N < b^{k+1}$ et donc $k = \lfloor \log_b(N) \rfloor$. Pour représenter N en base b, on a besoin de $\lfloor \log_b(N) \rfloor + 1$ chiffres.

Définition 4.1. *La taille d'un entier $N \neq 0$ est $\log_2(|N|)$ (c'est l'ordre de grandeur du nombre de bits qu'il faut pour le représenter).*

On peut distinguer le *coût en temps* et *le coût en mémoire*. On s'intéressera surtout au coût en temps.

Définition 4.2. *Le coût en temps ou complexité binaire d'un algorithme est le nombre d'opérations élémentaires effectuées au cours de cet algorithme. Il dépend de la taille des données.*

On considère un algorithme qui prend en entrée un entier $N \in \mathbb{N}^*$. On note $c(N)$ son coût en temps. On dit que l'algorithme est :

- *linéaire* si $c(N) =_{N \to +\infty} O\big(\log_2(N)\big)$,
- *quadratique* si $c(N) =_{N \to +\infty} O\big(\log_2(N)^2\big)$,
- *polynomial* si $c(N) =_{N \to +\infty} O\big(\log_2(N)^\alpha\big)$ pour un certain $\alpha \in \mathbb{N}$,
- *exponentiel* si $c(N) =_{N \to +\infty} O\big(2^{\log_2(N)^\alpha}\big)$ pour un certain $\alpha \in \mathbb{N}$.

On prendra garde au fait qu'un algorithme tel que $c(N) \sim N$ est exponentiel et non linéaire. Pour qu'un algorithme soit considéré comme réalisable en pratique, il faut que son coût soit polynomial (avec α pas trop grand).

4.1.2 Coût des opérations arithmétiques de base

Proposition 4.3. *Le coût en calcul de l'addition de deux entiers naturels m, n par l'algorithme usuel (en posant l'addition, dans une base $b \in \mathbb{N}_{\geq 2}$) ou de la soustraction (en la posant également) est $O\big(\max\big(\log_2 m, \log_2 n\big)\big)$ opération élémentaires.*

Démonstration. Soit $k = \lfloor \max\big(\log_b m, \log_b n\big) \rfloor$. On suppose que m et n sont représentés dans la base b par les listes $M = [a_k, \ldots, a_0]$ et $N = [a'_k, \ldots, a'_0]$, c'est à dire que $m = \sum_{i=0}^{k} a_i b^i$, $n = \sum_{i=0}^{k} a'_i b^i$ avec $a_i, a'_i \in [\![0, b-1]\!]$ pour tous $i \in [\![0, k]\!]$ (on complète M ou N avec des 0, afin d'obtenir deux listes de longueur $k+1$). On utilise l'algorithme suivant :

Algorithme addition(M, N)
$k \leftarrow \text{long}(M) - 1$
$i \leftarrow 0$
$R \leftarrow 0$
$S = [\,]$
Pour i allant 0 à k :
| $s \leftarrow M[k-i] + N[k-i] + R$
| si $s \geq b$:
|$R \leftarrow 1$
|$s \leftarrow s - b$
| sinon $s \geq b$:
|$R \leftarrow 0$
| $S \leftarrow [s] + S$
| $k \leftarrow k + 1$
si $R = 1$:
| $S \leftarrow [1] + S$
renvoyer S.

Cet algorithme calcule la somme de m et n (exercice le vérifier). Il y a $k+2$ étapes et chacune de ces étapes s'effectue en au plus 3 opérations élémentaires. Le cas de la soustraction se démontre similairement.

□

Proposition 4.4. *(exercice) Soient $m, n \in \mathbb{N}^*$.*

1. *Le coût en calcul du produit mn en posant la multiplication est $O\big(\log_2(m)\log_2(n)\big)$.*
2. *Si $m \leq n$, le coût en calcul de la division euclidienne (par l'algorithme vu en primaire) de m par n est $O\big(\log_2(|m/n|)\log_2(|n|)\big)$.*

Remarque 4.5. *Il existe des algorithmes plus efficaces pour calculer le produit de deux « grands » entiers (à partir de 1000 chiffres). Par exemple l'algorithme de Schöhnage-Strassen permet de calculer le produit de deux entiers de taille n en $O(n\log_2(n)\log_2(\log_2(n)))$.*

Soit $n \in \mathbb{N}^*$. On représente les éléments de $\mathbb{Z}/n\mathbb{Z}$ par des entiers de $[\![0, n-1]\!]$. On note $\pi : \mathbb{Z} \twoheadrightarrow \mathbb{Z}/n\mathbb{Z} \simeq [\![0, n-1]\!]$ la projection canonique (si $k \in \mathbb{Z}$, $\pi(k)$ est le reste de la division euclidienne de k par n). Par la proposition 4.4 que si $x \in \mathbb{Z}$, le coût du calcul de son image dans $\mathbb{Z}/n\mathbb{Z}$ est $O\big(\log_2(x/n)\log_2(n)\big)$. Si $x, y \in \mathbb{Z}/n\mathbb{Z}$, le coût du calcul de $x+y$ est $O\big(\log_2(n)\big)$. En effet, on écrit $x = \overline{a}$ et $y = \overline{b}$, avec $a, b \in [\![0, n-1]\!]$. Alors $x+y = \pi(a+b) \in \{a+b, a+b-n\}$ et on conclut avec la proposition 4.3. Le coût du calcul de xy est $O(\log_2(n)^2)$. En effet, on calcule $ab \in [\![0, (n-1)^2]\!]$, puis on effectue la division euclidienne de ab par n et on conclut avec la proposition 4.4.

4.2 Exponentiation rapide

Soit A un anneau, $x \in A$ et $n \in \mathbb{N}$. Pour calculer x^n, on peut utiliser l'algorithme « naïf » suivant :

Algorithme Puissance(x, n)
$k \leftarrow 0$
$b \leftarrow 1$
tant que $k < n$:
| $b \leftarrow b.x$
| $k \leftarrow k + 1$
renvoyer b.

On suppose que l'on dispose d'un algorithme `binaire`(n), qui prend en entrée un entier naturel n, et qui renvoie la liste $L = [a_k, \dots, a_0]$ d'éléments de $\{0, 1\}$ telle que $a_k \neq 0$, $a_i \in \{0, 1\}$ pour tous $i \in [\![0, k]\!]$ et $n = \sum_{i=0}^{k} n_i 2^i$. L'algorithme suivant calcule prend en entrée un élément x d'un monoïde et un entier naturel n et renvoie x^n.

Algorithme 1. expo-rapide(x,n)
$L \leftarrow$`binaire`(n)
$k \leftarrow \text{long}(L) - 1$
$P \leftarrow x$
$a \leftarrow 1$
pour i allant de 0 à k
| si $L[k - i] = 1$:
| | $a \leftarrow aP$
| $P \leftarrow P^2$
renvoyer a.

En effet, pour $i \in [\![0, k]\!]$, notons a_i, P_i, etc. les valeurs de a, P, etc. avant le passage i dans la boucle. On a $P_0 = x = x^{(2^0)}$ et $a_0 = 1$. Par récurrence, on a $P_i = x^{(2^i)}$, si $i \in [\![0, k]\!]$ et $a_i = \prod_{j<i|n_j=1} P_j = \prod_{j<i|n_j=1} x^{2^j} = x^{\sum_{j<i|n_j=1} 2^j} = x^{\sum_{j<i} n_j 2^j}$. Si on note a_{k+1} la valeur de a après le passage k dans la boucle, on a donc $a_{k+1} = x^{\sum_{j\leq i} n_j 2^j} = x^n$.

Pour calculer x, on a effectué $k+1$ élévations au carré (pour calculer les valeurs des P_i), et $k+1$ calculs de produits (pour calculer les valeurs des a_i). On a donc utilisé $O(\log_2(n))$ produits (les élévations au carré sont des produits), ce qui est beaucoup mieux que $O(n)$.

En comparaison avec l'algorithme « naïf », il y a donc beaucoup moins de multiplications à effectuer dans M ($O(\log_2(n))$ contre $O(n)$). Si $A = \mathbb{Z}$, le coût de calcul (temps de calcul) d'une multiplication $x.y$ augmente lorsque x et y augmentent. Il n'est donc pas évident a priori que le calcul de x^n soit plus rapide avec l'algorithme d'exponentiation rapide qu'avec l'algorithme naïf (car x^m croît avec m, si $x \geq 2$). Par contre, si $A = \mathbb{Z}/N\mathbb{Z}$, pour $N \in \mathbb{N}^*$ le coût du calcul de ab, pour $a, b \in \mathbb{Z}/N\mathbb{Z}$ est $O\big(\log_2(N)^2\big)$. Le coût du calcul de x^n pour $n \in \mathbb{N}$ par l'algorithme d'exponentiation rapide est donc $O\big(\log_2(N)^2 \log_2(n)\big)$ alors que celui par l'algorithme naïf est $O\big(\log_2(N)^2 n\big)$.

Remarque 4.6. *On peut aussi programmer l'exponentiation rapide de façon itérative en utilisant : pour $x \in M$ (où M est un monoïde) et $n \in \mathbb{N}$,*

$$x^n = \begin{cases} (x^{n/2})^2 & \text{si } n \text{ pair;} \\ x(x^{(n-1)/2})^2 & \text{si } n \text{ impair.} \end{cases}$$

Algorithme 1. *expo-rapide(x,n)*
si $n = 0$
| *renvoyer 1*
sinon
| *si* $n \ mod \ 2 = 0$
| | $r \leftarrow$ *expo-rapide*$(x, \frac{n}{2})$
| | *renvoyer* r^2
| *sinon*
| | $r \leftarrow$ *expo-rapide*$(x, \frac{n-1}{2})$
| | *renvoyer* $x \times r^2$.

On peut alors montrer que pour calculer x^n*, cet algorithme utilise* $O(\log_2(n))$ *produits dans* M *(exercice).*

Chapitre 5

Tests de primalité

5.1 Premiers algorithmes

Proposition 5.1. *Soit n un entier composé positif. Alors il a un diviseur premier $\leq \sqrt{n}$.*

Par contraposée, on a donc : si pour tout $p \in \mathbb{P}$ tel que $p \leq \sqrt{n}$ premier, p ne divise pas n, alors n est premier.

Démonstration. Soient $n \in \mathbb{N}$ ($n \geq 2$) composé et p son plus petit diviseur ≥ 2 (qui est premier par le lemme 2.13).

Alors p divise n donc il existe $k \in \mathbb{Z}$ tel que $n = kp$. Par définition, $k \geq p$, donc $pk = n \geq p^2$, d'où $p \leq \sqrt{n}$. □

Premiers tests de primalité

Premier test

Commençons par un test en supposant qu'on connaisse la liste de tous les nombres premiers jusqu'à $\sqrt{n}$, notée $L = [p_0, p_1, \ldots, p_{k-1}]$. (Par exemple, grâce à un algorithme qui, prenant en entrée n, renvoie tous les nombres premiers $\leq \sqrt{n}$.) L'algorithme ci-dessous renvoie si n est premier ou non et le plus petit diviseur premier si n est composé.

Algorithme 9. Primalité(n)
$i \leftarrow 0$
tant que n mod $L[i] \neq 0$ et $i \leq k$
$i \leftarrow i + 1$
si $i \leq k$
  | renvoyer « composé », $L[i]$
renvoyer premier

L'avantage est qu'on obtient le plus petit diviseur premier de n.

Nombre de divisions à effectuer : $O(\sqrt{n}/\ln n)$, d'après le théorème de Hadamard-La Vallée-Poussin (théorème 2.15).

Second test

On va maintenant donner un algorithme sans connaître les nombres premiers $\leq \sqrt{n}$. Cet algorithme teste 2 et tous les entiers impairs $\leq \sqrt{n}$ comme diviseur potentiel de n. On a les mêmes entrée et sortie que l'algorithme précédent.

Algorithme 10. Primalité2(n)

```
si n = 2
  | renvoyer premier
sinon
  | si n mod 2 = 0
  |   | renvoyer composé et 2
  | sinon
  |   | i ← 3
  |   | tant que n mod i = 0 et i ≤ √n
  |   |   | i ← i + 2
  |   | si i ≤ √n
  |   |   | renvoyer composé et i
  |   | sinon
  |   |   | renvoyer premier
```

L'avantage est qu'on obtient à nouveau le plus petit diviseur premier de n.
Nombre de divisions à effectuer : $O(\sqrt{n})$.

Crible d'Ératosthène

Soit $n \in \mathbb{N}^*$. On écrit dans un tableau les $n-1$ nombres de 2 à n. On conserve 2 puis on barre tous ses multiples stricts. Le premier nombre du tableau non barré est 3. On le conserve et on barre tous ses multiples stricts. Le premier nombre du tableau non barré est 5. On le conserve et on barre tous ses multiples stricts, et ainsi de suite. On fait ainsi apparaître les premiers nombres premiers $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, ... Lorsque l'on a a barré tous les multiples stricts de p_i pour $i \leq k$, le plus petit nombre supérieur à p_k qui n'est pas encore barré est p_{k+1}. Dés que l'on arrive à un nombre premier $p \geq n$, tous les nombres non barrés dans la tableau sont premiers et on a donc tous les nombres premiers inférieurs ou égaux à n. Cet algorithme donne une méthode simple pour déterminer tous les nombres premiers inférieurs à un entier positif donné : il ne nécessite aucun calcul, seulement le déplacement d'un curseur. Il utilise par contre une mémoire importante.

Exemple. *Pour $n = 30$:*

	②	③	~~4~~	⑤	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~

Nous allons maintenant introduire des nouveaux tests de primalité, beaucoup plus rapides, mais probabilistes. Ils se baseront sur des propriétés de $(\mathbb{Z}/n\mathbb{Z})^\times$, pour $n \in \mathbb{N}^*$.

5.2 Le groupe $(\mathbb{Z}/n\mathbb{Z})^\times$

5.2.1 Structure

Notation. $(\mathbb{Z}/n\mathbb{Z})^\times$ *désigne l'ensemble des éléments inversibles de* $\mathbb{Z}/n\mathbb{Z}$.

On rappelle que par la proposition 3.2, $(\mathbb{Z}/n\mathbb{Z})^\times = \{\overline{a} \in \mathbb{Z}/n\mathbb{Z} \mid a \wedge n = 1\}$.

Théorème 5.2. *Soit* $n \in \mathbb{N}^*$. *Alors les conditions suivantes sont équivalentes :*

1. n *est premier,*
2. $\mathbb{Z}/n\mathbb{Z}$ *est intègre,*
3. $\mathbb{Z}/n\mathbb{Z}$ *est un corps.*

Démonstration. (à connaître)

« (1) ⇒ (2) » Soient $\overline{x}, \overline{y} \in \mathbb{Z}/n\mathbb{Z}$ tels que $\overline{x}.\overline{y} = 0$. Alors $n \mid xy$, et n premier, donc $n \mid x$ ou $n \mid y$, i.e. $\overline{x} = 0$ ou $\overline{y} = 0$, par le lemme d'Euclide (lemme 2.12).

« (2) ⇒ (1) » Écrivons $n = n_1 n_2$ où $n_1, n_2 > 1$. On a $\overline{n} = \overline{n_1}.\overline{n_2} = 0$, et $\overline{n_1}, \overline{n_2} \neq 0$, donc $\mathbb{Z}/n\mathbb{Z}$ n'est pas intègre.

« (1) ⇒ (3) » Soit $\overline{x} \in \mathbb{Z}/n\mathbb{Z} \setminus \{0\}$. Alors n ne divise pas x et comme n est premier, $n \wedge x = 1$, donc $\overline{x}$ est inversible, par la proposition 3.2.

« (3) ⇒ (2) » Soient $x, y \in \mathbb{Z}/n\mathbb{Z}$ tels que $xy = 0$. Si $y \neq 0$, alors y est inversible, donc $x = xyy^{-1} = 0.y^{-1} = 0$, donc $\mathbb{Z}/n\mathbb{Z}$ est intègre.

□

Remarques 5.3.

- *Plus généralement, tout corps est un anneau intègre (par la preuve donnée dans le résultat précédent).*
- *(pour aller plus loin) Plus généralement, tout anneau intègre fini est un corps. En effet, soit* A *un tel anneau. Soit* $x \in A \setminus \{0\}$. *Soit* $m_x : A \to A$ *définie par* $m_x(y) = xy$, *pour* $y \in A$. *Alors comme* A *est intègre,* m_x *est injective, donc elle est surjective, donc il existe* $y \in A$ *tel que* $xy = 1$, *c'est à dire que* x *est inversible. On n'a en fait pas besoin de supposer* A *commutatif (exercice : démontrer que tout anneau fini sans diviseur de zéro est un corps non nécessairement commutatif). De plus, le théorème de Wedderburn (1905) affirme que tout corps fini est commutatif. On en déduit que tout anneau fini sans diviseur de* 0 *est un corps commutatif.*

Corollaire 5.4. *Soit* $p \in \mathbb{P}$. *Alors l'ensemble des solutions de l'équation* $\overline{x}^2 = \overline{1}$ *d'inconnue* $\overline{x} \in \mathbb{Z}/p\mathbb{Z}$ *est* $\{-\overline{1}, \overline{1}\}$. *Cet ensemble a* 2 *éléments, sauf si* $p = 2$.

Démonstration. Soit $x \in \mathbb{Z}$. Alors $\overline{x}^2 = \overline{1}$ si et seulement si $\overline{x}^2 - \overline{1} = (\overline{x} + \overline{1})(\overline{x} - \overline{1}) = \overline{0}$ si et seulement si $\overline{x} \in \{\overline{1}, \overline{-1}\}$. □

Remarque. *Dans* $\mathbb{Z}/35\mathbb{Z}$, $\overline{6}$ *est solution de l'équation* $\overline{x}^2 \equiv 1[35]$ *donc le corollaire est faux dans le cas général.*

Théorème 5.5. *1. Soit* $n \in \mathbb{N}^*$. *Alors* $(\mathbb{Z}/n\mathbb{Z})^\times$ *est un groupe abélien. Son cardinal, noté* $\varphi(n)$, *est appelé indicatrice d'Euler. On a* $\varphi(n) = |\{j \in [\![0, n-1]\!] \mid j \wedge n = 1\}|$.

2. *Si $n = n_1 n_2$, avec n_1, n_2 premiers entre eux, alors $\varphi(n) = \varphi(n_1)\varphi(n_2)$.*
3. *Pour tout $p \in \mathbb{P}$ et $k \in \mathbb{N}^*$, on a $\varphi(p) = p - 1$ et $\varphi(p^k) = p^{k-1}(p-1)$. Plus généralement, pour tout $n \in \mathbb{N}^*$, on a $\varphi(n) = n \prod_{p \in \mathbb{P}, p|n}(1 - 1/p)$.*

Démonstration.

(1) exercice.

(2) (à connaître) Par le théorème des restes chinois, il existe un isomorphisme de $\phi : \mathbb{Z}/n\mathbb{Z} \xrightarrow{\sim} \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$. Alors ϕ envoie tout élément inversible sur un élément inversible (car si $xy = 1$, $\phi(xy) = \phi(1) = 1$) et de même pour ϕ^{-1}, donc $\phi\big((\mathbb{Z}/n\mathbb{Z})^\times\big) = (\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z})^\times$. Soit $(\overline{x}, \overline{y}) \in \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$. Alors (x, y) est inversible si et seulement si il existe $(x', y') \in \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$ tel que $(xx', yy') = (1 \bmod n_1, 1 \bmod n_2)$, si et seulement si x et y sont inversibles (et alors $(x, y)^{-1} = (x^{-1}, y^{-1})$). On a donc $(\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z})^\times = (\mathbb{Z}/n_1\mathbb{Z})^\times \times (\mathbb{Z}/n_2\mathbb{Z})^\times$ et en prenant les cardinaux, on en déduit que $\varphi(n_1 n_2) = \varphi(n_1)\varphi(n_2)$.

(3) (à connaître) On a $\varphi(p^k) = |\{j \in [\![0, p^k - 1]\!] \mid j \wedge p^k = 1\}|$. Soit $j \in [\![0, p^k - 1]\!]$. Alors $j \wedge p^k = 1$ si et seulement si $j \wedge p = 1$ si et seulement si p ne divise pas j. Ainsi

$$\{j \in [\![0, p^k - 1]\!] \mid j \wedge p^k = 1\} = [\![0, p^k - 1]\!] \setminus \{0, p, 2p, \ldots, p^k - p\} = [\![0, p^k - 1]\!] \setminus p[\![0, p^{k-1} - 1]\!],$$

et donc $\varphi(p^k) = p^{k-1}(p-1) = p^k(1 - \frac{1}{p})$. On conclut en utilisant (2).

□

5.2.2 Test de non-primalité de Fermat

Théorème 5.6 (Théorème d'Euler)**.** *Soient $n \in \mathbb{N}^*$ et $a \in \mathbb{Z}$. Si a est un entier premier avec n, alors $a^{\varphi(n)} \equiv 1[n]$.*

Démonstration. On a $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^\times$. L'ordre de $\overline{a}$ divise $|\mathbb{Z}/n\mathbb{Z}|^\times = \varphi(n)$, donc $\overline{a}^{\varphi(n)} = \overline{1}$. □

Corollaire 5.7 (Petit théorème de Fermat)**.** *Soit p un nombre premier. Alors si a est un entier non divisible par p, on a $a^{p-1} \equiv 1[p]$.*

Ce théorème fournit un autre test de non-primalité : si on peut trouver un entier a tel que $a \wedge n = 1$ et $a^{n-1} \not\equiv 1[n]$, alors n n'est pas premier.

Proposition 5.8. *(voir TD) Soit $n \in \mathbb{N}_{\geq 2}$. Supposons qu'il existe $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^\times$ tel que $\overline{a}^{n-1} \neq 1$. Alors $|\{\overline{a} \in \mathbb{Z}/n\mathbb{Z} \mid \overline{a}^{n-1} \neq \overline{1}\}| \geq \frac{n}{2}$.*

On peut donc envisager le test de primalité suivant. On choisit $k \in \mathbb{N}$.

Si $a, b \in \mathbb{N}$, on note Hasard(a, b) une fonction renvoyant un entier choisi aléatoirement entre a et b.

Algorithme 12. Test-Fermat(n, k)
pour i de 1 à k :
| $a \leftarrow$ Hasard$(1, n-1)$
| si $(a \bmod n)^{n-1} \neq 1 \bmod n$:
| | renvoyer « n est composé »
renvoyer « n est peut-être premier »

Par la proposition précédente, si Test-Fermat(n, k) renvoie « n est peut-être premier », alors la probabilité qu'il existe $\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^\times$ tel que $\overline{a}^{n-1} \neq \overline{1}$ est inférieure ou égale à $\frac{1}{2^k}$. Par exemple si $k = 30$, cette probabilité est inférieure à 1 sur 1 milliard. Le problème de ce test est qu'il existe des nombres dits de Carmichael.

Définition 5.9. *Soient $a, n \in \mathbb{N}^*$. On dit que n est un nombre **pseudo-premier de Fermat de base** a si n est composé et si*

$$a^{n-1} \equiv 1[n]. \quad (*)$$

*On dit que n est un **nombre de Carmichael** (ou un nombre pseudo-premier de Fermat) si n est composé et pour tous $a \in \mathbb{Z}$ tels que $a \wedge n = 1$, on a $a^{n-1} \equiv 1[n]$.*

Il existe une infinité de nombre de Carmichael (c'est un théorème de Alford, Grandville et Pomerance de 1994). Le plus petit nombre de Carmichael est $561 = 3 \times 11 \times 17$. L'ensemble des nombres de Carmichael est assez mal compris. Empiriquement il y en a beaucoup moins que de nombre premiers. Il y a par exemple 22 nombres de Carmichael entre 1 et 10 000 alors qu'il y a 1 229 nombres premiers entre 1 et 10 000. Un nombre de 50 bits (respectivement 100) choisi aléatoirement a moins d'une chance sur 10^6 (respectivement 10^{13}) de faire échouer le test. Par rapport aux tests présentés précédemment, il est beaucoup plus rapide. Si on fixe k (par exemple $k = 30$), alors en utilisant l'exponentiation rapide, on a un coût en $O(k \log(n)^3)$ opérations élémentaires, alors que les tests que l'on a vus précédemment ont un coût supérieur à $\sqrt{n}/\ln(n)$ opérations élémentaires. En revanche, lorsque le test renvoie que n est composé, il ne fournit pas de diviseur de n. En pratique ce test n'est pas très utile, car il existe des tests à peine plus sophistiqués (les tests de Miller-Rabin et de Solovay-Strassen par exemple) pour lesquels il n'existe pas de nombre « pseudo-premiers ».

5.3 Test de Miller-Rabin

L'algorithme se base sur la propriété suivante :

Proposition 5.10. *Soit $p \geq 3$ premier. Soient $s \in \mathbb{N}^*$ et d impair tels que $p - 1 = 2^s d$. Alors pour tout $a \in [\![1, p-1]\!]$, on a :*

- *soit $a^d \equiv 1[p]$,*
- *soit il existe $r \in [\![0, s-1]\!]$ tel que $a^{2^r d} \equiv -1[p]$.*

Démonstration. (à connaître) On se place dans $\mathbb{Z}/p\mathbb{Z}$. Supposons que $\overline{a}^d \neq \overline{1}$. Par le théorème de Fermat, $\overline{a}^{2^s d} = \overline{1}$. Soit $r = \min\{r' \in [\![1, s]\!] \mid \overline{a}^{2^{r'} d} = \overline{1}\} - 1$. Alors $\overline{a}^{2^r d} \neq 1$ et $(\overline{a}^{2^r d})^2 = (\overline{a})^{2^{r+1} d} = \overline{1}$, par définition de r. Par le corollaire 5.4, on en déduit que $\overline{a}^{2^r d} = \overline{-1}$. □

Test témoin

Définition 5.11. *Soit $n \in \mathbb{N}^*$ impair. Écrivons $n - 1 = 2^s d$, avec $s \in \mathbb{N}^*$ et d impair. Un* ***témoin de Miller*** *pour n est un nombre $a \in [\![1, n-1]\!]$ tel que $a^d \not\equiv 1[n]$ et pour tout $r \in [\![0, s-1]\!]$, $a^{2^r d} \not\equiv -1[n]$.*

Le test suivant renvoie vrai si a est un témoin de la nature composée de n, à savoir $a^{n-1} \not\equiv 1[n]$.

Algorithme 13.

```
Témoin(a, n)
si Euclide(a, n) ≠ 1
   | renvoyer vrai
sinon
   | s ← 0
   | d ← n − 1
   | tant que d mod 2 = 0
   |    | s ← s + 1
   |    | d ← d/2
   | x ← a^d mod n
   | si x = 1 mod  ou x = −1 mod n
   |    | renvoyer faux
   | tant que s ≠ 1
   |    | x ← x × x mod n
   |    | si x = n − 1
   |    |    | renvoyer faux
   |    | s ← s − 1
renvoyer vrai
```

Remarque. *Si Témoin(a, n) renvoie vrai, cela signifie que n est composé. Sinon, on ne sait pas, et dans ce cas, n est soit premier soit pseudo-premier de base a.*

Théorème 5.12. *(Rabin, admis) Soit $n \in \mathbb{N}_{\geq 9}$. Si n est composé, alors au moins les trois quarts des $a \in [\![2, n-2]\!]$ sont des témoins de Miller.*

On en déduit le test probabiliste suivant qui prend en entrée un entier $n \geq 9$ et un entier k (tel que $1/4^k$ soit « petit »).

Algorithme 14. Miller-Rabin(n, k)
pour j de 1 à k
| $a \leftarrow$ Hasard$(1, n-1)$
| si Témoin(a, n)
| | renvoyer composé
renvoyer premier ?

Remarque. *Le test peut fonctionner pour un nombre de Carmichael. Par exemple, pour $n = 561$, on peut remarquer que Témoin$(7, 561)$ renvoie vrai car $n - 1 = 35 \times 2^4$ et : $7^{35} \equiv 241[561]$, $7^{70} \equiv 298[561]$, $7^{140} \equiv 166[561]$ et $7^{280} \equiv 67[561]$. Donc si 7 est choisi pour tester la primalité de 561, le test de Miller-Rabin fonctionne.*

Il est certain que si le test de Miller-Rabin renvoie composé, alors n est vraiment composé, mais s'il renvoie « premier ? », cela signifie qu'on n'a pas trouvé de témoin de la nature composé de n, mais cela ne veut pas dire pour autant que n est forcément premier. Cependant, si n est composé, la probabilité que le test renvoie « premier ? » est inférieure à $\frac{1}{4^k}$.

5.4 Test de Solovay-Strassen

5.4.1 Symbole de Legendre

Définition 5.13 (Symbole de Legendre)**.** *Soient p un nombre premier impair et $a \in \mathbb{Z}$. On définit alors $(\frac{a}{p})$ par*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{si } p \mid a \\ 1 & \text{si } p \nmid a \text{ et si } \exists x \in \mathbb{Z} : x^2 \equiv a[p] \\ -1 & \text{sinon} \end{cases}$$

Remarque. *$(\frac{a}{p})$ ne dépend que de la classe de a modulo p.*

Théorème 5.14. *(admis) Soit $p \in \mathbb{P}$. Alors $(\mathbb{Z}/p\mathbb{Z})^\times$ est cyclique. Autrement dit, $(\mathbb{Z}/p\mathbb{Z})^\times \simeq \mathbb{Z}/(p-1)\mathbb{Z}$.*

Proposition 5.15. *(critère d'Euler) Soient p un nombre premier impair et $a \in \mathbb{Z}$. Alors*

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}}[p]$$

Démonstration.

- $p \mid a$: c'est clair.
- Si $p \nmid a$, alors $a \bmod p \in (\mathbb{Z}/p\mathbb{Z})^\times$. Soit g un générateur de $(\mathbb{Z}/p\mathbb{Z})^\times$ (ce qui existe par le théorème 5.14). Alors $(\mathbb{Z}/p\mathbb{Z})^\times = \{g^i, 0 \leq i \leq p-2\}$. Alors $(g^{\frac{p-1}{2}})^2 = g^{p-1} = \overline{1}$ et $g^{\frac{p-1}{2}} \neq \overline{1}$ donc $g^{\frac{p-1}{2}} = \overline{-1}$ (par le corollaire 5.4). Donc si $a = g^k$, alors $a^{\frac{p-1}{2}} = (\overline{-1})^k$. De plus, g^k est un carré dans $(\mathbb{Z}/p\mathbb{Z})^\times$ si et seulement si k est pair. D'où le résultat.

□

Remarque. *Voici une autre démonstration de cette proposition, plus élémentaire. Si $x \in (\mathbb{Z}/p\mathbb{Z})^\times$ est un carré, écrivons $x = y^2$, où $y \in \mathbb{Z}/p\mathbb{Z}$. Alors par le théorème de Fermat, $x^{(p-1)/2} = y^{p-1} = \overline{1}$. Ainsi, tout carré non nul est racine du polynôme $X^{(p-1)/2} - 1 \in \mathbb{Z}/p\mathbb{Z}[X]$. De plus, il y a exactement $(p-1)/2$ carrés dans $(\mathbb{Z}/p\mathbb{Z})^\times$ (exercice, le démontrer). Comme $\mathbb{Z}/p\mathbb{Z}$ est intègre, $X^{(p-1)/2} - 1$ admet au plus $(p-1)/2$ racines donc il admet exactement $(p-1)/2$ racines, qui sont les carrés de $(\mathbb{Z}/p\mathbb{Z})^\times$.*

Corollaire 5.16. *Soient p un nombre premier impair et $(a, b) \in \mathbb{Z}^2$. Alors*

$$\Big(\frac{ab}{p}\Big) = \Big(\frac{a}{p}\Big)\Big(\frac{b}{p}\Big).$$

5.4.2 Loi de réciprocité quadratique

Théorème 5.17 (Loi de réciprocité quadratique (admis))**.** *Soient p et q deux nombres premiers impairs distincts. Alors*

$$\Big(\frac{p}{q}\Big) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}\Big(\frac{q}{p}\Big).$$

Première loi complémentaire : $\Big(\frac{-1}{p}\Big) = (-1)^{(p-1)/2}$.

Deuxième loi complémentaire (admis) : $\Big(\frac{2}{p}\Big) = (-1)^{(p^2-1)/2}$.

La loi de réciprocité quadratique permet de calculer rapidement des symboles de Legendre. Par exemple, pour calculer $\Big(\frac{35}{61}\Big)$, on a $\Big(\frac{35}{61}\Big) = \Big(\frac{5}{61}\Big)\Big(\frac{7}{61}\Big)$. On a

$$\Big(\frac{5}{61}\Big) = (-1)^{(5-1)/2.(61-1)/2}\Big(\frac{61}{5}\Big) = \Big(\frac{61}{5}\Big) = \Big(\frac{1}{5}\Big) = 1.$$

On a

$$\Big(\frac{7}{61}\Big) = (-1)^{(7-1)/2.(61-1)/2}\Big(\frac{61}{7}\Big) = \Big(\frac{5}{7}\Big) = \Big(\frac{-2}{7}\Big) = \Big(\frac{-1}{7}\Big)\Big(\frac{2}{7}\Big) = (-1)^{(7-1)/2}.(-1)^{(7^2-1)/8} = -1.$$

On a donc $\Big(\frac{35}{61}\Big) = -1$, donc 35 n'est pas un carré modulo 61.

Remarque 5.18. *La notation $(-1)^{\frac{p-1}{2}\frac{q-1}{2}}$ permet d'être concis. Néanmoins, pour calculer $(-1)^{\frac{p-1}{2}\frac{q-1}{2}}$, il n'est vraiment optimal de calculer $\frac{p-1}{2}\frac{q-1}{2}$. En fait $(-1)^{\frac{p-1}{2}\frac{q-1}{2}} = -1$ si et seulement si $\frac{p-1}{2}\frac{q-1}{2}$ est impair, si et seulement si $(p-1)/2$ et $(q-1)/2$ sont impairs. Comme p et q sont impairs, $(p-1)/2$ est pair si et seulement si $p \equiv 1[4]$ et $(p-1)/2$ est impair si et seulement si $p \equiv 3[4]$.*

— *Même remarque pour $(-1)^{(p^2-1)/8}$. Il vaut souvent mieux calculer p modulo 8 plutôt que de calculer $(p^2-1)/8$. En effet, si $p \equiv \pm 1[8]$, on peut écrire $p = 8k \pm 1$, où $k \in \mathbb{Z}$. Alors $p^2 = (8k \pm 1)^2 = 64k^2 \pm 16k + 1$ donc $(p^2-1)/8 = 8k^2 \pm 2k$ est pair. De même, si $p \equiv \pm 3[8]$, $(p^2-1)/8$ est impair.*

5.4.3 Symbole de Jacobi

Définition 5.19. *[Symbole de Jacobi] Soit n un entier naturel impair. On écrit $n = \prod_{1\leq j\leq k} p_j^{\alpha_j}$ où les p_j sont des nombres premiers (impairs). Pour $a \in \mathbb{Z}$, on définit le symbole de Jacobi $\left(\frac{a}{n}\right)$ par*

$$\left(\frac{a}{n}\right) = \prod_{j=1}^{k} \left(\frac{a}{p_j}\right)^{\alpha_j} \in \{-1, 0, 1\}.$$

Remarque.

1. *Si n est premier, les symboles de Legendre et de Jacobi se confondent.*
2. *Si $a, n \in \mathbb{N}$, avec n impair sont tels que $\left(\frac{a}{n}\right) = -1$, alors a n'est pas un carré modulo n. En effet, dans ce cas là , il existe un diviseur premier p de n tel que $\left(\frac{a}{p}\right) = -1$. Alors a n'est pas un carré modulo p. Ainsi pour tout $b, k \in \mathbb{Z}$, $b^2 + kp \neq a$ et donc $b^2 + kn = b^2 + kp\frac{n}{p} \neq a$. Par contre, on peut avoir $\left(\frac{a}{n}\right) = 1$, sans que a soit un carré modulo n. Par exemple $\left(\frac{2}{9}\right) = \left(\frac{2}{3}\right)^2 = 1$, mais 2 n'est pas un carré modulo 9.*
3. *La définition de $\left(\frac{a}{n}\right)$ que l'on vient de voir utilise la décomposition de n en produit de facteurs premier. On va voir une méthode pour calculer $\left(\frac{a}{n}\right)$ « rapidement », sans factoriser n, ce qui permettra d'utiliser ce symbole dans un test de primalité.*

Proposition 5.20. *Soient $m, m_1, m_2, n, n_1, n_2 \in \mathbb{Z}$, avec n, n_1, n_2 impairs. Alors :*

1. $\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$.
2. $\left(\frac{m}{n_1 n_2}\right) = \left(\frac{m}{n_1}\right)\left(\frac{m}{n_2}\right)$.
3. $\left(\frac{m}{n}\right) = 0 \iff m \wedge n \neq 1$.
4. *Soit r le reste dans la division euclidienne de m par n. Alors $\left(\frac{m}{n}\right) = \left(\frac{r}{n}\right)$.*
5. *Si m est impair, $\left(\frac{m}{n}\right) = (-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}\left(\frac{n}{m}\right)$.*
6. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$; $\qquad \left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$

Démonstration. (1-4) : exercice, (pour 4) on pourra utiliser le fait que $r \in a - p_j\mathbb{Z}$ pour tout $j \in [\![1, k]\!]$, avec la notation de définition 5.19).

(5) Écrivons $m = \prod_{i=1}^{k} p_i$ et $n = \prod_{j=1}^{\ell} q_j$, où $k, \ell \in \mathbb{N}$, $p_1, \ldots, p_k, q_1, \ldots, q_\ell \in \mathbb{P}_{\geq 3}$ (on

ne suppose pas les p_i, q_j distincts). Alors,

$$\begin{aligned}\left(\frac{m}{n}\right) &= \prod_{i=1}^{k}\prod_{j=1}^{\ell}\left(\frac{p_i}{q_j}\right)\\ &= \prod_{i=1}^{k}\prod_{j=1}^{\ell}(-1)^{(p_i-1)/2.(q_j-1)/2}\left(\frac{q_j}{p_i}\right), \text{ d'après la loi de réciprocité quadratique,}\\ &= \left(\prod_{i=1}^{k}\prod_{j=1}^{\ell}\left(\frac{q_j}{p_i}\right)\right)\left(\prod_{i=1}^{k}\prod_{j=1}^{\ell}(-1)^{(p_1-1)/2.(q_j-1)/2}\right)\\ &= \left(\frac{n}{m}\right)\left(\prod_{i=1}^{k}\prod_{j=1}^{\ell}(-1)^{(p_1-1)/2.(q_j-1)/2}\right).\end{aligned}$$

Il reste à montrer que

$$\prod_{i=1}^{k}\prod_{j=1}^{\ell}(-1)^{(p_i-1)/2.(q_j-1)/2} = (-1)^{(m-1)/2.(n-1)/2} \tag{5.1}$$

On a $(-1)^{(m-1)/2.(n-1)/2} = -1$ si et seulement si $(m-1)/2.(n-1)/2$ est impair si et seulement si $(m-1)/2$ et $(n-1)/2$ sont impairs donc

$$(-1)^{(m-1)/2.(n-1)/2} = -1 \Leftrightarrow m \equiv n \equiv 3[4]. \tag{5.2}$$

Comme les p_i sont impairs, on a $p_i \equiv 1[4]$ ou $p_i \equiv 3[4]$, pour $i \in [\![1,k]\!]$. Posons $X_p = \{i \in [\![1,k]\!] \mid p_i \equiv 3[4]\}$. On a alors

$$m = \prod_{i=1}^{k} p_i \equiv \prod_{i\in X_p} p_i \equiv (-1)^{|X_p|}[4]. \tag{5.3}$$

De même, si $X_q = \{j \in [\![1,\ell]\!] \mid q_i \equiv 3[4]\}$, $n \equiv (-1)^{|X_q|}[4]$.

En appliquant (5.2) à p_i et q_j au lieu de m et n, on obtient $\prod_{i=1}^{k}\prod_{j=1}^{\ell}(-1)^{(p_i-1)/2.(q_j-1)/2} = (-1)^{|X_p|\times|X_q|}$.

On a donc $\prod_{i=1}^{k}\prod_{j=1}^{\ell}(-1)^{(p_i-1)/2.(q_j-1)/2} = -1$ si et seulement si $|X_p|.|X_q|$ est impair, si et seulement si $|X_p|$ et $|X_q|$ sont impairs, d'où l'égalité (5.1), d'après (5.2) et (5.3). On en déduit (5).

(6) Avec les mêmes notations qu'en (5), on a $\left(\frac{-1}{n}\right) = \prod_{i=1}^{\ell}\left(\frac{-1}{q_j}\right) = \prod_{j=1}^{\ell}(-1)^{(q_j-1)/2} = (-1)^{|X_q|} = (-1)^{(n-1)/2}$.

On a $\left(\frac{2}{n}\right) = \prod_{i=1}^{\ell}\left(\frac{2}{q_i}\right)$. Soient $E = \{i \in [\![1,\ell \mid q_i \equiv \pm 1[8]\}$, $F_1 = \{i \in [\![1,\ell]\!] \mid q_i \equiv 3[8]\}$ et $F_2 = \{i \in [\![1,\ell]\!] \mid q_i \equiv -3[8]\}$. D'après la remarque 5.18, on a $\prod_{i=1}^{\ell}\left(\frac{2}{q_i}\right) \equiv (-1)^{|F_1|+|F_2|}$. Il faut également déterminer n mod 8. Dans $\mathbb{Z}/8\mathbb{Z}$, on a

$$\overline{n} = \prod_{i=1}^{\ell}\overline{q_i} = \left(\prod_{i\in E}\overline{q_i}\right)\left(\prod_{i\in F_1}\overline{q_i}\right)\left(\prod_{i\in F_2}\overline{q_i}\right).$$

On a $\prod_{i\in E}\overline{p_i} = \pm\overline{1}$. On a $\left(\prod_{i\in F_1}\overline{q_i}\right) = \overline{3^{|F_1|}}$ et $\left(\prod_{i\in F_2}\overline{q_i}\right) = \pm\overline{3^{|F_2|}}$. On a donc $\overline{n} = \pm\overline{3}^{|F_1|+|F_2|}$ et comme $\overline{3}^2 = \overline{1}$, on obtient $\overline{n} = \pm\overline{3}^{|F_1|+|F_2| \bmod 2}$. D'après la remarque 5.18, on a donc $(-1)^{(n^2-1)/8} = (-1)^{|F_1|+|F_2|}$.

□

Exemple. 1. *Déterminons* $\left(\frac{33}{45}\right)$. *On a* $\left(\frac{33}{45}\right) = (-1)^{(45-1)/2.(33-1)/2}\left(\frac{45}{33}\right) = \left(\frac{45}{33}\right)$. *On a* $\left(\frac{45}{33}\right) = \left(\frac{12}{33}\right) = \left(\frac{4}{33}\right)\left(\frac{3}{33}\right)$. *Comme* 3 *divise* 33, *on en déduit que* $\left(\frac{33}{45}\right) = 0$.

2. *Déterminons* $\left(\frac{123}{259}\right)$. *On a* $\left(\frac{123}{259}\right) = (-1)^{(123-1)/2.(259-1)/2}\left(\frac{259}{123}\right) = -\left(\frac{259}{123}\right) = \left(\frac{13}{123}\right)$. *On a* $\left(\frac{13}{123}\right) = (-1)^{(13-1)/2.(123-1)/2}\left(\frac{123}{13}\right) = \left(\frac{6}{13}\right)$. *On a* $\left(\frac{6}{13}\right) = \left(\frac{2}{13}\right)\left(\frac{3}{13}\right)$. *Comme* $13 \equiv 3[8]$, $\left(\frac{2}{13}\right) = -1$. *On a* $\left(\frac{3}{13}\right) = (-1)^{(13-1)/2.(3-1)/2}\left(\frac{13}{3}\right) = \left(\frac{1}{3}\right) = 1$. *On a donc* $\left(\frac{123}{259}\right) = -1$.

L'algorithme suivant permet de calculer le symbole de Jacobi. Il prend en entrée un entier m et un entier impair positif n et renvoie $\left(\frac{m}{n}\right)$.

Algorithme 15.

```
Jacobi(m, n)
si n = 1
renvoyer 1
sinon
  | a ← m mod n
  | si a = 0
  |    | renvoyer 0
  | sinon
  |    | si a mod 2= 0
  |    |    |si n%8 in [1, 7] :
  |    |    |   |Renvoyer Jacobi(a/2, n)
  |    |    |sinon :
  |    |    |   | Renvoyer −Jacobi(a/2, n) :
  |    | sinon
  |    |   | si a%4 = 1 ou n%4 = 1 :
  |    |   |    | Renvoyer Jacobi(n, a)
  |    |   | sinon :
  |    |   |    | Renvoyer −Jacobi(n, a)
```

Montrons par récurrence sur n que l'algorithme 15 termine.

- Si $n = 1$, OK.
- Supposons que l'algorithme 15 termine pour Jacobi(m', n') où $m' \in \mathbb{Z}$ et n' impair $< n$. On peut supposer sans perte de généralité que $m = a$ où $0 < a < n$.
  Si a est impair, on calcule récursivement Jacobi(n, a) qui termine bien par hypothèse de récurrence car $a < n$.
  Si a est pair, on a $a = 2^k b$ où $k \geq 1$ et b impair. L'algorithme se ramené à Jacobi(b, n) qui termine car b impair.
  Pour vérifier que le résultat renvoyé par Jacobi(a, n) est le bon, on utilise les propriétés précédentes.

5.4.4 Le test

L'algorithme suivant est un nouveau test de primalité, fondé sur le symbole de Jacobi. On utilise à nouveau la fonction Hasard(a, b). Il prend en entrée un entier naturel impair ≥ 3.

Théorème 5.21. *Soit n un entier impair ≥ 3. Alors :*

1. *n est premier si et seulement si pour tous $a \in \mathbb{Z}$ tel que $a \wedge n = 1$, $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2}[n]$,*
2. *si n est composé, $\{a \in [\![2, n-1]\!] \mid \ a^{(n-1)/2} \equiv \left(\frac{a}{n}\right)[n]\}$ a au plus $\varphi(n)/2$ éléments.*

Démonstration. La preuve de (1) qui peut-être faite en utilisant les propriétés des nombres de Carmichael est admise.

(2) On se place dans $(\mathbb{Z}/n\mathbb{Z})^\times$. Alors $G := \{\overline{a} \in (\mathbb{Z}/n\mathbb{Z})^\times \mid \overline{a}^{(n-1)/2} = \overline{\frac{a}{n}}\}$ est un sous-groupe de $(\mathbb{Z}/n\mathbb{Z})^\times$. Par (1), on sait que G est un sous groupe strict de $(\mathbb{Z}/n\mathbb{Z})^\times$. Par le théorème de Lagrange, $|G|$ divise $|(\mathbb{Z}/n\mathbb{Z})^\times| = \varphi(n)$, d'où le résultat. □

Algorithme : Solovay-Strassen(n)
$a \leftarrow$ Hasard$(1, n-1)$
$j \leftarrow$ Jacobi(a, n)
si $j = 0$
  | renvoyer composé
sinon
  | $b \leftarrow a^{\frac{n-1}{2}} \bmod n$
  | si $b = n-1$
  | | $b \leftarrow -1$
  | si $b \neq j$
  | | renvoyer composé
  | sinon
  | | renvoyer premier ?

Remarque. *Si n est composé, la probabilité de tomber sur un a montrant que n est composé est $\geq 1/2$.*
Supposons que n est composé. Si on fait le test k fois, où les k entiers sont choisis indépendamment entre 1 et $n-1$, la probabilité de prouver que n est composé est $\geq 1 - 1/2^k$.

Chapitre 6

Un peu de cryptographie

6.1 Chiffrement de César et variations

Historique. Le mot cryptographie vient du grec *kryptos* (caché) et de *graphein* (écriture) : il s'agit de l'art de transformer un message pour tenter de le rendre illisible par toute autre personne que son destinataire.

Exemple. *Le code de Jules César : il s'agit d'écrire un message en remplaçant chaque lettre par celle située trois rangs après elle dans l'ordre alphabétique.*

- L'algorithme de cryptage est appelé **chiffre de substitution**.
- Le destinataire a la **clé** du chiffrement, i.e. l'information lui permettant d'effectuer le **déchiffrement**.

Exercice. *Déchiffrer YHQLYLGLYLFL, chiffré par le biais du code de César.*

Un code plus efficace serait de fabriquer un chiffre de substitution à partir d'un mot ou d'une suite de mots.

Exemple. *Le texte choisi dans cet exemple est LE ROI AGAMEMNON.*
On retire les lettres répétitives et on colle les mots : on obtient LEROIAGMN, qui forment alors les lettres du début de notre alphabet chiffré ; les autres lettres suivent normalement. Nous obtenons alors :

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

Exercice. *Déchiffrer SIZTLBMZRIZBLTCZLUB.*

Remarque. *Si un indiscret tombe sur le message précédent, même s'il n'y a que dix lettres différentes, s'il n'a pas la clé, il devra envisager tous les arrangements possibles : il y en a 26!/17!, c'est-à -dire plus de 19000 milliards.*

L'analyse de la fréquence des lettres dans l'alphabet permet de « casser » tous ces codes pour les textes suffisamment longs. La fréquence des lettres dans l'alphabet en langue française (elle dépend de l'échantillon utilisé) est la suivante :

E	S	A	I	N	T	R	U	L	O
14,69%	8,01%	7,54%	7,18%	6,89%	6,88%	6,49%	6,12%	5,63%	5,29%

.

On peut alors utiliser la méthode suivante. On cherche la lettre qui apparaît le plus dans le texte chiffré. Cela devrait être un E. Les deux lettres suivantes qui apparaissent le plus devraient être un S et un A etc. On peut faire un chiffrement par blocs, c'est à dire choisir un nombre k, et coder des blocs de k lettres. C'est le chiffrement de Vigenére. Cependant, il est aussi vulnérable à une analyse statistique de fréquence.

6.2 Chiffrement symétrique et asymétrique

Détails

Alice veut envoyer un message à Bob. Didier, un importun, aimerait bien parvenir à lire ce message.

But. *Pour Alice et Bob : envoyer des messages entre eux sans que personne d'autre ne puisse les lire.*

Alice chiffre alors son message et l'envoie à Bob de manière à ce que seul Bob sache le déchiffrer. Il faut donc que :

- Bob sache lire le message d'Alice ;
- Didier ne sache pas le lire.

Notations.

- M=$\{messages\}$
- F_A *: fonction de chiffrement d'Alice.*
- F_B *: fonction de déchiffrement de Bob.*
- m *: message d'origine (non chiffré).*
- $c = F_A(m)$ *: message codé.*

On a donc $m = F_B(c)$, donc F_B est la fonction inverse de F_A, i.e. $F_B \circ F_A = Id$.

$$c \longrightarrow m$$

Une première idée serait que Alice et Bob partagent un secret qui leur permet de chiffrer et de déchiffrer les messages : on parle dans ce cas de **chiffrement symétrique**.

Le problème de ce chiffrement est qu'Alice et Bob doivent être en mesure d'échanger ce secret en toute sécurité.

Exemple : le masque jetable ou chiffrement de Vernam

On choisit une clé (une suite de caractères, par exemple des entiers de 0 à 25) $c = (c_0, \ldots, c_k)$ de manière « aléatoire » au moins aussi longue que le message $m = (m_0, \ldots, m_k)$

à chiffrer. Le message envoyé est alors $m' = (m_0 + c_0 \% 26, \ldots, m_k + c_k \% 26)$. Pour le destinataire du message, il suffit alors de calculer $(m' - c)\%26$. En théorie, ce chiffrement est incassable. En pratique, il est peu utilisé, car il nécessite de pouvoir échanger une clé secrète longue avant de pouvoir communiquer.

Une seconde idée serait alors de faire un **chiffrement asymétrique**.

Principe.
- *Bob possède une clé secrète. Il ne la confie à personne, pas même à Alice.*
- *Il fabrique, à partir de cette clé secrète, une clé qu'il rend publique, de manière à ce qu'Alice puisse la consulter.*

Cette clé publique doit être construite de manière à ce que :
- Alice sache chiffrer des messages à l'aide de la clé publique.
- Bob sache déchiffrer des messages à l'aide de sa clé secrète.
- Didier ne sache pas déchiffrer les messages sans la clé ni deviner la clé secrète à partir de la clé publique et du message chiffré c.

6.3 Systéme RSA

Ce système a été mis en place par Rivest, Schamir et Adleman ; il s'agit d'un système asymétrique ou encore d'un système à clé publique.

6.3.1 Construction des clés publique et privée

Principe.
- *Bob choisit deux nombres premiers distincts p et q de taille comparable.*
- *Bob calcule $n = pq$ et $\varphi = (p-1)(q-1)$.*
- *Bob choisit un entier e qui vérifie $1 < e < \varphi$ et $e \wedge \varphi = 1$.*
- *Bob cherche l'unique entier $2 \leq d \leq \varphi - 1$ tel que $de \equiv 1[\varphi]$.*
- *On obtient les clés publique et privée souhaitées :*
  - ⋆ *clé publique : (n, e) ;*
  - ⋆ *clé secrète : (d, φ).*

Plusieurs questions se posent alors :
1) Comment calculer d ?
2) Peut-on calculer d à partir de (n, e) ?

Réponses. 1) *On applique Euclide-étendu(e, φ) qui renvoie $(1, x, y)$; alors $d = x$.*
2) *Pour calculer d, il suffit de connaître $\varphi(n) = (p-1)(q-1)$, et donc de connaître la décomposition de n en produit de facteurs premiers.*

6.3.2 Chiffrement

Principe.

- *Alice récupére la clé publique de Bob.*
- *Alice écrit le message sous la forme d'un entier* $0 \leq m \leq n-1$.
- *Alice calcule* $C = m^e \ mod \ n$.
- *Alice envoie* C *à Bob.*

Remarques.

- *Le calcul de* C *s'effectue à l'aide de l'algorithme d'exponentiation modulaire.*
- *Pour retrouver* m *à l'aide de* C *et de la clé publique* (n, e) *, il faut soit retrouver* d *(ce qui a été vu à la partie précédente), soit calculer une racine e-ième de* C *mod* n*, i.e. trouver* m_1 *tel que* $m_1^e \equiv C[n]$*, ce qui est, de façon calculatoire, très difficile si on ne connaît pas la décomposition en produit de facteurs premiers de* n*. Si on connaît cette décomposition, on calcule* d *pour obtenir la racine.*

6.3.3 Déchiffrement

Principe. *Bob utilise la clé privée* (d, φ) *pour retrouver* $m = C^d \ mod \ n$*. Ceci se base sur le résultat suivant :*

Lemme 6.1. *Soit* $m \in \mathbb{Z}/n\mathbb{Z}$*. Alors* $m^{ed} = m$.

Démonstration. Soit $\phi : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ l'isomorphisme chinois. Écrivons $\phi(\overline{m}) = (m_p, m_q)$. Soit $i \in \{p, q\}$. Si $m_i = \overline{0}$, alors $m_i^{ed} = 0 = m_i$. Si $m_i \neq \overline{0}$, alors $m_i^{i-1} = \overline{1}$, par le théorème de Fermat. Comme $i - 1$ divise φ, qui divise $ed - 1$, on en déduit que $m_i^{ed-1} = \overline{1}$ et donc $m_i^{ed} = m_i$. On a $m_i^{ed} = m_i$, dans les deux cas. On en déduit que $m^{ed} = (m_p^{ed}, m_q^{ed}) = (m_p, m_q)$, d'où le résultat. □

6.4 Sur la sécurité de RSA

Cette partie utilise des résultats du texte public 2018 C2 sur la page agreg.org.

Montrons que trouver la clé secrète d uniquement à l'aide de la clé publique (n, e) est calculatoirement équivalent à factoriser n.

- Si on sait factoriser $n = pq$, on peut alors calculer $\varphi = (p-1)(q-1)$; puis d tel que $1 < d < \varphi$ et $ed \equiv 1[\varphi]$.
- Si on connaît n, e et d, alors comme $ed \equiv 1[\varphi]$, il existe $k \in \mathbb{Z}$ tel que $1 = ed + k\varphi$, donc si on choisit $a \in (\mathbb{Z}/n\mathbb{Z})^\times$, on a $a^{ed-1} \equiv 1[n]$. Décomposons à présent $ed - 1$ sous la forme $ed - 1 = 2^s t$ où $s \in \mathbb{Z}$ et t impair.

Lemme 6.2. *Soit* $a \in (\mathbb{Z}/n\mathbb{Z})^\times$*. Alors la probabilité que* $a^t \neq 1$ *et qu'il existe* $j \in [\![1, s]\!]$ *tel que* $a^{2^{j-1}s} \neq \pm\overline{1}$ *et* $a^{2^j s} = \overline{1}$ *est supérieure ou égale à* $\frac{3}{8}$.

Démonstration. Soit $X = \{a \in (\mathbb{Z}/n\mathbb{Z})^\times \mid a^t = \overline{1}\}$. Soient $\phi : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ l'isomorphisme chinois et $X' = \phi(X) = \{(x_1, x_2) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \mid (x_1^t, x_2^t) = (\overline{1}, \overline{1})\}$. Soit $a = (a_1, a_2) \in X'$. Alors $O(a) := \{a, (a_1, -a_2), (-a_1, a_2), -a\}$ vérifie $O(a) \cap X = \{a\}$. En effet soit $\epsilon_1, \epsilon_2 \in \{-1, 1\}$. Alors $(\epsilon_1 a_1, \epsilon_2 a_2)^t = (\epsilon_1 a_1^t, \epsilon_2 a_2^t) = (\overline{\epsilon_1}, \overline{\epsilon_2})$, par hypothèse sur a. Le seul choix permettant d'obtenir $(\epsilon_1 a_1, \epsilon_2 a_2) \in X'$ est donc $\epsilon_1 = \epsilon_2 = 1$.

De plus, comme $X' \subset (\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z})^\times = \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/(q-1)\mathbb{Z}$, $\{a, -a, \pm(a_1, -a_2)\} = 4$. Soit maintenant $b = (b_1, b_2) \in X$. Supposons que $O(a) \cap O(b) \neq \emptyset$. Alors il existe $\epsilon_1, \epsilon_2, \epsilon_1', \epsilon_2' \in \{-1, 1\}$ tel que $(\epsilon_1 a_1, \epsilon_2 a_2) = (\epsilon_1' b_1, \epsilon_2' b_2)$. On a donc $a = (\epsilon_1 \epsilon_1' b_1, \epsilon_2 \epsilon_2' b_2)$. Comme $a^t = b^t = (\overline{1}, \overline{1})$ et que t est impair, on en déduit que $\epsilon_1 \epsilon_1' = 1$ et $\epsilon_2 \epsilon_2' = 1$ et donc $\epsilon_1 = \epsilon_1'$ et $\epsilon_2 = \epsilon_2'$. Donc $a = b$. On a donc $\bigsqcup_{a \in X'} O(a) \subset \phi\big((\mathbb{Z}/n\mathbb{Z})^\times\big)$. En prenant les cardinaux, on obtient que $4|X'| = 4|X| \leq |\phi\big((\mathbb{Z}/n\mathbb{Z})^\times\big)| = |(\mathbb{Z}/n\mathbb{Z})^\times|$. Soit maintenant $Y = (\mathbb{Z}/n\mathbb{Z})^\times \setminus X$. Soit $Z = \{a \in Y \mid \exists j \in [\![1, s]\!] \mid a^{2^{j-1}} \neq \pm\overline{1} \text{ et } a^{2^j} = \overline{1}\}$. On verra en TD que $|Z| \geq \frac{|Y|}{2}$. On a donc $|Z| \geq \frac{|Y|}{2} = \frac{\varphi(n) - |X|}{2} \geq \frac{\varphi(n) - 1/4\varphi(n)}{2} = \frac{3\varphi(n)}{8}$. □

Soient x et j deux entiers tels que $a = \overline{x}$ et j soient comme dans le lemme 6.2. Alors $(x^{2^{j-1}t} - 1) \wedge n$ est un facteur non trivial de n. En effet, $\overline{x^{2^j t} - 1} = \overline{(x^{2^{j-1}t} - 1)(x^{2^{j-1}t} + 1)} = \overline{0}$ et $\overline{(x^{2^{j-1}t} - 1)}, \overline{(x^{2^{j-1}t} + 1)} \neq \overline{0}$ dans $\mathbb{Z}/n\mathbb{Z}$. On en déduit que $\overline{x^{2^j t} - 1}$ et $\overline{x^{2^j t} + 1}$ ne sont pas inversibles dans $\mathbb{Z}/n\mathbb{Z}$ et donc que $(x^{2^{j-1}t} + 1)$ et $(x^{2^{j-1}t} - 1)$ ne sont pas premiers avec n.
Il suffit alors de choisir aléatoirement des entiers x jusqu'à ce qu'on obtienne un facteur non trivial de n pour factoriser n.

Bibliographie

[Cha94] Jean-Luc Chabert. Histoire d'algorithmes : du caillou à la puce. Belin, 1994.

[CLRS02] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction à l'algorithmique. 2002.

[Com98] François Combes. Algèbre et géométrie, volume 51. Bréal, 1998.

[Dem08] Michel Demazure. Cours d'algèbre, volume 1 of Nouvelle Bibliothèque Mathématique [New Mathematics Library]. Cassini, Paris, 2008. Primalité. Divisibilité. Codes. [Primality. Divisibility. Codes].

[MKVOV96] Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. Handbook of applied cryptography. CRC press, 1996.

[Pic99] Philippe Saux Picart. Cours de calcul formel : Algorithmes fondamentaux. 1999.