CHAPITRE 1

ARITHMÉTIQUE DES ENTIERS ET COMPLEXITÉ

D'un point de vue effectif, tous les protocoles cryptographiques que nous étudierons se réduisent en dernière analyse à la manipulation d'entiers (ou de familles d'entiers) et à la détermination explicite d'opérations arithmétiques entre eux. La conception de nombreux cryptosystèmes se base le plus souvent sur la relative simplicité de certaines de ces opérations (effectuées en phase de cryptage, qui se doit d'être rapide) ou, au contraire, de leur difficulté (ce qui rend difficile le déchiffrement lors de la transmission du message chiffré). Nous allons commencer par un rapide rappel des porpriétés algébriques et arithmétique des entiers. Ces constructions et résultats, essentiellement contenus dans les ***Éléments*** d'Euclide, sont présentés dans le langage des anneaux, ce qui permet, entre autre, de se familiariser avec ce formalisme et de passer en revue quelques notions de base d'algèbre commutative. Nous étudierons ensuite l'aspect effectif des opérations, en introduisant la notion de complexité.

1.1. Rappels sur l'arithmétique des entiers

1.1.1. Structure d'anneau euclidien — La plupart des propriétés arithmétiques et algébriques des entiers, ainsi que l'effectivité de nombreux algorithmes découlent de l'existence d'une ***division euclidienne***, ce qui munit $\mathbb{Z}$ d'une structure d'***anneau euclidien***, une classe d'anneaux particulièrement adaptés à des constructions effectives. Le résultat ci-dessous nous accompagne depuis les bancs du collège (voire plus loin).

Théorème 1.1.1 — *Étant donnés deux entiers a et b, avec b non nul, il existe un unique couple d'entiers q et r, appelés respectivement* ***quotient*** *et* ***reste*** *de la division euclidienne de a par b, tels que*

$$a = bq + r,$$

avec $0 \leq r < |b|$.

Démonstration — L'ensemble

$$A = \{a - bq \mid q \in \mathbb{Z}\} \cap \mathbb{N}$$

est non vide (car il contient l'élément $a + |ba| \geq 0$) et possède donc un plus petit élément r. Par construction, on a l'identité $a = bq + r$, avec $q \in \mathbb{Z}$. Si l'on avait $r \geq |b|$, on obtiendrait les relations

$$r' = r - |b| = a - b(q \pm 1) \in A$$

et $r' < r$, ce qui est exclu. On a donc les inégalités $0 \leq r < |b|$. Finalement, si $qb + r = q'b + r'$, avec $0 \leq r, r' < |b|$, on obtient l'identité $b(q - q') = r' - r$. Pour $q \neq q'$, on en déduit les relations

$$|b| \leq |b(q' - q)| = |r - r'| < |b|,$$

ce qui est une fois encore exclu. On a donc $q = q'$ et, par suite $r = r'$, d'où l'unicité du quotient et du reste. □

D'un point de vue pratique, lorsque b est positif, en considérant la division euclidienne $a = bq + r$, on a l'identité $q = \lfloor a/b \rfloor$, où $\lfloor x \rfloor$ désigne la ***partie entière*** d'un réel x, i.e. le plus grand entier n tel que $n \leq x$.

Remarque 1.1.2 — En procédant comme dans la démonstration ci-dessus, on montre l'existence et l'unicité d'un couple d'entiers q et r tels que $a = bq + r$, avec cette fois $-|b|/2 \leq r < |b|/2$. Dans ce cas, on parle de ***division euclidienne modifiée***.

1.1.2. Idéaux — On rappelle qu'un idéal $\mathfrak{a}$ d'un anneau A est ***monogène*** s'il existe un élément $a \in A$, appelé ***générateur*** de $\mathfrak{a}$, tel que

$$\mathfrak{a} = aA = \{ab \mid b \in A\},$$

Le générateur d'un idéal monogène n'est généralement pas unique. Deux éléments a et b de A sont ***associés*** s'ils engendrent le même idéal, i.e. si $aA = bA$.

Lemme 1.1.3 — *Deux éléments $a, b \in A$ d'un anneau intègre A sont associés si et seulement s'il existe un élément $u \in A^\times$ tel que $a = ub$.*

Démonstration — L'assertion étant immédiate lorsque $ab = 0$, supposons a et b non nuls. Si $aA = bA$, on a en particulier $b \in aA$ et $a \in bA$, ou encore $b = ua$ et $a = vb$, avec $u, v \in A$, d'où les identités $a = bv = auv$ et, par suite, $a(1 - uv) = 0$. L'anneau A étant intègre et a étant non nul, on a alors $1 - uv = 0$, ce qui implique que u est inversible, d'inverse v. Réciproquement, si $b = ua$, avec $u \in A^\times$, on a $b \in aA$ et $a = u^{-1}b \in bA$, d'où les inclusions $bA \subset aA$ et $aA \subset bA$, qui sont alors des égalités. □

Un anneau A est ***principal*** s'il est intègre et si tous ses idéaux sont monogènes.

Théorème 1.1.4 — *L'anneau $\mathbb{Z}$ est principal.*

Démonstration — L'anneau $\mathbb{Z}$ étant intègre, il suffit de vérifier que tout idéal $\mathfrak{a}$ de $\mathbb{Z}$ est monogène. L'assertion étant immédiate pour $\mathfrak{a} = 0$, on supose $\mathfrak{a}$ non nul. L'ensemble

$$\mathfrak{a}^+ = \{a \in \mathfrak{a} \mid a > 0\} \subset \mathbb{N}$$

est non vide (car $\mathfrak{a}$ possède un élément non nul a, auquel cas $|a| = \pm a \in \mathfrak{a}^+$) et possède donc un élément minimal $n > 0$. On a on alors l'inclusion $n\mathbb{Z} \subset \mathfrak{a}$. Réciproquement étant donné $m \in \mathfrak{a}$, en considérant la division euclidienne $m = nq + r$, avec $0 \leq r < n$, on obtient les relations $r = m - nq \in \mathfrak{a}$. Pour $r > 0$, on obtiendrait $r \in \mathfrak{a}^+$, contredisant la minimalité de n. On a donc $r = 0$, d'où $m = nq \in n\mathbb{Z}$ et, par suite, l'inclusion $\mathfrak{a} \subset n\mathbb{Z}$, qui est alors une égalité. □

Proposition 1.1.5 — *Un idéal de $\mathbb{Z}$ possède un unique générateur positif. En particulier, l'application qui associe à un entier naturel n l'idéal $n\mathbb{Z}$ définit une bijection entre $\mathbb{N}$ et l'ensemble des idéaux de $\mathbb{Z}$.*

Démonstration — L'identité $\mathbb{Z} = \{\pm 1\}$ combinée avec le lemme 1.1.3 affirme qu'un idéal de $\mathbb{Z}$, qui est monogène (cf. le théorème 1.1.4), possède au plus deux générateurs, opposés l'un de l'autre (qui coïncident si et seulement si l'idéal est nul), un seul d'entre eux étant positif. □

1.1.3. Divisibilité, plus grand diviseur commun, identité de Bézout — De manière générale, si a et b sont deux éléments d'un anneau A, on dit que b ***divise*** a, ou que a est un ***multiple*** de b s'il existe $c \in A$ tel que $a = bc$. On utilise la notation $b|a$ pour indiquer que b divise a. Les idéaux sont l'outil le plus adapté dans toute question liée à la divisibilité. Le résultat élémentaire suivant en est une première illustration.

Lemme 1.1.6 — *Étant donnés deux éléments a et b d'un anneau A, on a $b|a$ si et seulement si $aA \subset bA$.*

Démonstration — Si $b|a$, on a $a = bc$, avec $c \in A$, d'où $a \in bA$ et, par suite, l'inclusion $aA \subset bA$. Réciproquement, pour $aA \subset bA$, on a $a \in aA \subset bA$, d'où $a = bc$, avec $c \in A$. □

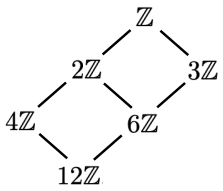
Revenant aux entiers, le résultat ci-dessous complète la proposition 1.1.5.

Proposition 1.1.7 — *Il existe une bijection naturelle entre l'ensemble des diviseurs positifs d'un entier n et l'ensemble des idéaux de $\mathbb{Z}$ contenant $n\mathbb{Z}$.*

Démonstration — Il suffit de combiner la proposition 1.1.5 et le lemme 1.1.6 □

Remarque 1.1.8 — La divisibilité définit une relation d'ordre sur $\mathbb{N}$ (mais pas sur $\mathbb{Z}$). Par ailleurs, l'ensemble des idéaux de $\mathbb{Z}$ est naturellement ordonné par l'inclusion (on pose ici $\mathfrak{a} \leq \mathfrak{b}$ si et seulement si $\mathfrak{b} \subset \mathfrak{a}$). Le résultat ci-dessus affirme que l'application qui associe à un entier naturel n l'idéal $n\mathbb{Z}$ est un isomorphisme d'ensembles ordonnés, i.e. une bijection qui est compatible avec les relations d'ordre.

Exemple 1.1.9 — Les six idéaux de $\mathbb{Z}$ contenant $12\mathbb{Z}$ sont $\mathbb{Z}, 2\mathbb{Z}, 3\mathbb{Z}, 4\mathbb{Z}, 6\mathbb{Z}$ et $12\mathbb{Z}$. La relation d'inclusion entre eux, qui correspond au treillis des diviseurs de 12 est décrite schématiquement dans la figure ci-dessous.



On rappelle que si $\mathfrak{a}$ et $\mathfrak{b}$ sont deux idéaux d'un anneau A, leur ***somme***

$$\mathfrak{a} + \mathfrak{b} = \{a + b \mid a \in \mathfrak{a}, b \in \mathfrak{b}\},$$

est également un idéal de A, c'est d'ailleurs le plus petit idéal contenant $\mathfrak{a} \cup \mathfrak{b}$. En ce qui concerne les entiers, étant donnés $a, b \in \mathbb{Z}$, la somme

$$a\mathbb{Z} + b\mathbb{Z} = \{ax + by \mid x, y \in \mathbb{Z}\}$$

des idéaux $a\mathbb{Z}$ et $b\mathbb{Z}$ est monogène (cf. le théorème 1.1.4), engendré par un unique entier naturel (cf. la proposition 1.1.5), appelé ***plus grand diviseur commun***, ou simplement ***pgcd*** de a et b. On le note généralement (a, b). Par constructon, il existe un couple d'entiers x et y vérifiant la relation

$$(a, b) = ax + by,$$

appelée ***identité de Bézout***.

Exercice 1.1.10 — Soient a et b deux entiers. Montrer qu'étant donné un entier naturel d, les conditions suivantes sont équivalentes :

1. On a $d = (a, b)$.
2. L'entier d divise a et b et si $c \in \mathbb{Z}$ est un diviseur commun à a et b alors c divise d (cette propriété justifie le nom de plus grand diviseur commun).

Deux entiers a et b sont ***premiers entre eux*** si $(a, b) = 1$, ce qui revient à affirmer qu'il existe une identité de Bézout $ax + by = 1$.

Exercice 1.1.11 — Soient a, b et c des entiers, avec $(a, b) = 1$. Montrer le ***lemme de Gauss*** : si a divise bc alors a divise c. En déduir que si $a|c$ et $b|c$ alors $ab|c$.

Le ***plus petit commun multiple***, ou ***ppcm*** de a et b, noté $[a, b]$ est l'unique générateur positif de l'idéal $a\mathbb{Z} \cap b\mathbb{Z}$.

Exercice 1.1.12 — Soient a et b deux entiers. Montrer qu'étant donné un entier naturel m, les conditions suivantes sont équivalentes :

1. On a $m = [a, b]$.

2. L'entier m est un multiple de a et b et si $c \in \mathbb{Z}$ est multiple commun à a et b alors c est un multiple de m (cette propriété justifie le nom de plus petit commun multiple).

De manière générale, étant donnés des entiers $a_1, \ldots, a_n$, leur pgcd, noté $(a_1, \ldots, a_n)$ est l'unique générateur positif de l'idéal $a_1\mathbb{Z} + \cdots + a_n\mathbb{Z}$. De même, leur ppcm est l'unique générateur positif de l'idéal $a_1\mathbb{Z} \cap \cdots \cap a_n\mathbb{Z}$.

Remarque 1.1.13 — Les définitions de pgcd et de ppcm proposées dans ce paragraphe ainsi que leurs propriétés s'appliquent en fait à tout anneau principal, si ce n'est que dans le cas général, on ne dispose pas nécessairement d'un générateur privilégié d'un idéal (on parle alors d'un pgcd plutôt que du pgcd de a et b).

Exercice 1.1.14 — Montrer qu'étant donnés des entiers a, b et c, on a les propriétés suivantes :

1. $(a, b + ac) = (a, b)$.
2. $(ab, ac) = |a|\,(b, c)$.
3. $(a, b)\,[a, b] = |ab|$.

1.1.4. Factorisation unique — Un ***nombre premier*** est un entier $n > 1$ tel que ses seuls diviseurs (positifs) sont 1 et n lui-même.

Proposition 1.1.15 — *Un entier naturel $n > 1$ est un nombre premier si et seulement si l'idéal $n\mathbb{Z}$ est maximal. En particulier, on en déduit le* ***lemme d'Euclide****, qui affirme que si un nombre premier divise le produit d'entiers alors il divise l'un des facteurs.*

Démonstration — La première assertion est une conséquence directe de la proposition 1.1.7. La seconde découle du fait que tout idéal maximal d'un anneau est premier (cf. l'appendice). □

Théorème 1.1.16 — *Un entier non nul n s'écrit de manière unique comme produit*

$$n = \pm \prod_p p^{e_p},$$

où p parcourt l'ensemble des nombres premiers, les entiers naturels e_p étant presque tous nuls (i.e. tous sauf un nombre fini d'entre eux).

Démonstration — On commence par montrer que tout nombre entier n possède une telle écriture. On procède par récurrence sur $|n|$: l'assertion étant claire pour $n = 1$ (auquel cas il suffit de poser $e_p = 0$ pour tout p), supposons $|n| > 1$ et la propriété vraie pour tout entier m tel que $|m| < |n|$.Si $|n|$ est premier, l'assertion est clairement remplie. Si n est composé, on a $n = ab$, avec $1 < a, b < n$ et il suffit

d'appliquer l'hypothèse de récurrence. Concernant l'unicité, considérons deux écritures $n = \pm\prod_p p^{e_p} = \pm\prod_p p^{f_p}$ et supposons qu'il existe un nombre premier q tel que $e_q \neq f_q$. On peut se réduire au cas $e_q < f_q$, ce qui amène à l'identité

$$\prod_{p\neq q} p^{e_p} = \pm q^{f_q - e_q} \prod_{p\neq q} p^{f_p}.$$

En particulier, le nombre premier q divise $\prod_{p\neq q} p^{e_p}$ et coïncide donc avec l'un des facteurs, ce qui est exclu. On a donc $e_p = f_p$ pour tout nombre premier p et le signe dans les deux expressions est clairement le même, d'où l'unicité. □

Remarque 1.1.17 — L'écriture (unique) décrite dans le résultat ci-dessus est en fait valable pour tout rationnel non nul x, si ce n'est que les exposants e_p peuvent être négatifs. Dans le langage des groupes, le théorème fondamental de l'arithmétique établit un isomorphisme entre $\mathbb{Q}^\times$ et $\mu_2 \times \bigoplus_p \mathbb{Z}$, où l'on a posé $\mu_2 = \mathbb{Z}^\times = \{\pm 1\}$.

Exercice 1.1.18 — Étant donnés deux entiers non nuls a et b, considérons leurs factorisations $a = \pm\prod_p p^{e_p}$ et $b = \pm\prod_p p^{f_p}$. Montrer que l'on a $a|b$ si et seulement si $e_p \leq f_p$ pour tout nombre premier p. En déduire les identités

$$\begin{cases} (a,b) = \prod_p p^{\min\{e_p, f_p\}}, \\ \\ [a,b] = \prod_p p^{\max\{e_p, f_p\}}. \end{cases}$$

1.1.5. L'algorithme d'Euclide étendu — Le calcul de pgcd est omniprésent en cryptographie algébrique. Il est par conséquent nécessaire de fournir un algorithme efficace permettant de le déterminer explicitement. Soient donc a et b deux entiers, que l'on suppose non nuls (dans le cas contraire, la calcul de leur pgcd est trivial). Le pgcd ne dépendant pas du signe des entiers considérés, on peut se réduire au cas $0 < b < a$ (pour $a = b$, on a clairement $(a,b) = a$). Tout d'abord, on serait tenté de considérer les factorisations de a et b en produit de nombres premiers, utilisant ensuite l'expression du pgcd de l'exercice 1.1.18. Ne disposant à ce jour d'aucun algorithme de factorisation réellement performant, cette démarche est à ecarter. Nous allons décrire une seconde méthode, le célèbre ***algorithme d'Euclide (étendu)***, qui s'avère bien plus efficace. On construit une suite finie d'entiers naturels $(r_i)_{i\geq 0}$, appelée ***suite des restes*** (associée à a et b) par le procédé suivant :

- On pose $r_0 = a$ et $r_1 = b$.
- Pour $i \geq 1$, si $r_i = 0$ le procédé s'arrête, sinon r_{i+1} est le reste d'une division euclidienne de r_{i-1} par r_i.

Il existe alors un unique entier $n \geq 1$ tel que $v(r_0) \geq v(r_1) > \cdots > v(r_n)$ et $r_{n+1} = 0$. Pour tout $i \in \{1, \ldots, n\}$, en posant

$$r_{i-i} = q_i r_i + r_{i+1},$$

on obtient la ***suite des quotients*** $(q_i)_{1\leq i\leq n}$. Finalement, on considère deux autres suites $(u_i)_{0\leq i\leq n}$ et $(v_i)_{0\leq i\leq n}$ définies par

$$\begin{cases} u_0 = 1, \\ u_1 = 0, \\ u_{i+1} = u_{i-1} - u_i q_i \text{ pour } i > 0, \end{cases} \quad \text{et} \quad \begin{cases} v_0 = 0, \\ v_1 = 1, \\ v_{i+1} = v_{i-1} - v_i q_i \text{ pour } i > 0. \end{cases}$$

Il peut être commode de présenter les étapes de calculs sous la forme du tableau suivant

	q_1	q_2	$\cdots$	q_{n-1}	q_n	
a	b	r_2	$\cdots$	r_{n-1}	r_n	0
1	0	u_2	$\cdots$	u_{n-1}	u_n	
0	1	v_2	$\cdots$	v_{n-1}	v_n	

Proposition 1.1.19 — *Avec les notations et hypothèses ci-dessus, on a les identités*

$$(a, b) = r_n = au_n + bv_n.$$

Démonstration — Les relations $(a, b) = (r_0, r_1)$ et $r_n = (r_n, r_{n+1})$ sont immédiatement vérifiées. Pour tout $i \in \{1, \ldots, n\}$, le point 1 de l'exercice 1.1.14 amène alors aux identités

$$(r_{i-1}, r_i) = (r_{i-1} - q_i r_i, r_i) = (r_{i+1}, r_i)\,,$$

d'où la première égalité. Concernant la seconde, montrons par récurrence que pour tout $i \in \{0, \ldots, n\}$, on a la relation $r_i = au_i + bv_i$. L'assertion étant trivialement vérifiée pour $i = 0$ et $i = 1$, supposons qu'elle est vraie pour un entier $i \geq 1$. On a alors les relations

$$\begin{aligned} r_{i+1} = r_{i-1} - q_i r_i &= au_{i-1} + bv_{i-1} - q_i(au_i + bv_i) = \\ &= a(u_{i-1} - q_i u_i) + b(v_{i-1} - q_i v_i) = au_{i+1} + bv_{i+1}, \end{aligned}$$

ce qui conclut la démonstration. □

Remarque 1.1.20 — Dans ce procédé, on peut librement remplacer la division euclidienne usuelle par sa version modifiée (cf. la remarque 1.1.2). Dans ce cas, les termes de la suite des restes (r_i) n'étant pas n'écessairement positifs, on obtient l'identité $|r_n| = (a, b)$ (le nombre d'itérations n n'étant généralement pas le même qu'avec la division euclidienne usuelle). Par contre, l'expression $r_n = au_n + bv_n$ reste valable. La suite $(|r_i|)$ est strictement décroissante, car pour $i > 0$, on a $|r_{i+1}| \leq |r_i|/2$, d'où la relation $|r_i| \leq |b|/2^{i-1}$. En particulier, on obtient $n \leq \log_2(|b|) + 1$, ce qui permet de quantifier le nombre d'itération nécessaires pour que l'algorithme d'Euclide étendu aboutisse (dans sa version modifiée). Une analyse similaire, faisant intervenir la ***suite de Fibonacci***, amène à une majoration semblable du nombre d'itérations nécessaires avec l'agorithme usuel (non modifié).

Exemple 1.1.21 — Déterminons explicitement le pgcd des entiers 1071 et 2023. On a le tableau

	1	1	8	
2023	1071	952	119	0
1	0	1	-1	
0	1	-1	2	

On en déduit la relation $(1071, 2023) = 119$ et l'identité de Bézout

$$2 \cdot 1071 - 1 \cdot 2023 = 119.$$

La division euclidienne modifiée amène au second tableau

	2	-9	
2023	1071	-119	0
1	0	1	
0	1	-2	

L'algorithme est donc plus rapide dans sa version modifiée.

1.2. Quelques notions de complexité

Ayant passé en revue les propriétés essentielles des entiers qui seront utilisés dans la suite du cours, on s'interesse à présent à l'aspect effectif des opérations et constructions présentées (somme, produit, division euclildienne, algorithme d'Euclide étendu,...).

1.2.1. Écriture d'un entier en base b — Nous allons brièvement rappeler une notion qui remonte à nos souvenirs de jeunesse : la numération en base b.

Théorème 1.2.1 — *Soit $b > 1$ un entier. Tout entier naturel n s'écrit de manière unique sous la forme*

$$n = a_0 + a_1 b + \cdots + a_k b^k + \cdots ,$$

où les entiers $a_0, a_2, \cdots \in \{0, \ldots, b-1\}$ sont presque tous nuls (i.e. nuls à partir d'un certain rang). Une telle expression est appelée ***écriture en base b de n****.*

Démonstration — Concernant l'existence, on procède par récurrence sur l'entier n : pour $n < b$, c'est immédiat. Soit donc $n \geq b$ un entier et supposons que tout entier $m < n$ possède une telle écriture. En particulier, en considérant la division euclidienne $n = bm + a_0$, avec $0 \leq a_0 < b$, on a $m < n$ d'où l'écriture $m = a_1 + a_2 b + a_3 b^2 + \cdots$, ce qui donne

$$n = a_0 + b(a_1 + a_2 b + \cdots) = a_0 + a_1 b + a_2 b^2 + \ldots .$$

Concernant l'unicité, il suffit de remarquer que a_0 est le reste de la division euclidienne de $n_0 = n$ par b ; il est donc unique. Dans ce cas a_1 est le reste de la division euclidienne

de $n_1 = (n_0 - a_0)/b$ par b, qui est également unique. En itérant ce procédé, on en déduit qu'il en est de même pour a_i, qui est le reste de la division euclidienne de $n_i = (n_{i-1} - a_{i-1})/b$ par b. □

En suivant la notation et les hypothèses du théorème 1.2.1, notons $\ell = \ell_b(n)$ le plus grand entier naturel tel que $a_\ell \neq 0$. On écrit alors $n = (a_\ell \cdots a_0)_b$ pour indiquer l'écriture en base b de n. Si aucune confusion n'est possible, on utilise également la notation $n = a_\ell \cdots a_0$.

Exercice 1.2.2 — Montrer que $\ell_b(a)$ est le plus petit entier ℓ tel que $b^{\ell+1} > a$, ce qui se traduit par l'identité

$$\ell_b(a) = \lfloor \log_b(a) \rfloor,$$

où $\lfloor x \rfloor$ désigne la partie entière d'un réel x.

Exemple 1.2.3 — Le tableau ci-dessous contient les écritures de l'entier 2024 en base $b < 10$.

2	3	4	5	6	7	8	9
11111101000	2202222	133220	31044	13212	5621	3750	2688

Exercice 1.2.4 — Soit $b > 1$ un entier. Montrer qu'un entier relatif n s'écrit de manière unique sous la forme

$$n = a_0 - a_1 b + a_2 b^2 + \cdots + a_k (-b)^k + \cdots ,$$

où les entiers $a_0, \ldots, a_k \in \{0, \ldots, b-1\}$ sont presque tous nuls. Déterminer une telle expression pour $b = 2$ et $n = 2024$.

Exercice 1.2.5 — Considérons un polynôme $f \in \mathbb{Z}[X]$ et supposons que tous ses coefficients sont des entiers naturels. Montrer que la donnée des entiers $n = f(1)$ et $m = f(n+1)$ permet de déterminer f.

1.2.2. La taille d'un entier en informatique — Ayant commencé à compter en utilisant les dix doigts de nos mains, c'est l'écriture en base 10 qui s'est imposée au fil des siècles. Cependant, d'un point de vue pratique, et particulièrement en ce qui concerne les application en informatique, on utilise le plus souvent l'écriture en base 2. Dans ce contexte, un ***bit*** est une variable qui ne peut prendre que les valeurs 0 et 1. En considérant son écriture en base 2, un entier naturel peut être assimilé à une suite de bits. De manière plus précise, étant donnée l'écriture $n = (a_\ell \ldots a_0)_2$, l'entier naturel $k = \ell_2(n) + 1$ est la ***taille*** de n, i.e. le nombre de bits nécéssaires pour le mémoriser. On dit alotrs que n est un ***entier de k bits***. On rappelle que l'on a l'identité $k = \lfloor \log_2(n) \rfloor + 1$ (cf. l'exercice 1.2.2). En particulier, n est un entier à ℓ bits si et seulement si l'on a l'encadrement $2^{k-1} \leq n < 2^k$. Le bit a_ℓ est le ***plus significatif*** et a_0 est le ***moins significatif***.

Exemple 1.2.6 — L'entier $2024 = (11111101000)_2$ est de taille 11. À ce jour, dans les applications cryptographiques tels que le cryptosystème RSA (que nous verrons dans le deuxième chapitre), on utilise des entiers de 1024 bits, ce qui correspond à 309 chiffres décimales.

Dans la pratique, on fixe la taille (maximale) k des entiers naturels considérés i.e. la capacité de l'ordinateur avec sur lequel on envisage d'implémenter un algorithme. Si $n = (a_\ell \cdots a_0)_2$ est l'écriture en base 2 d'un tel entier, on a alors $\ell \leq k-1$ et $(0, \ldots, 0, a_\ell, \ldots, a_0)$ est le k-plet de bits correspondant à n (on complète par des 0). Lors de l'implémentation de tout algorithme, il faut impérativement tenir compte de la possibilité de dépassement de cette capacité, amenant à une erreur (***overflow*** en anglais).

Remarque 1.2.7 — Voulant encoder un entier relatif quelconque, on peut considérer un bit supplémentaire correspondant à son signe. Il est important de signaler que dans la pratique, on utilise plutôt la technique du ***complément à deux***, qui permet d'effectuer des soustractions de manière plus naturelle. Voulant éviter de s'éloigner des objectifs du cours, nous ne détaillerons pas cette méthode.

1.2.3. Opérations élémentaires sur les bits et opérations arithmétiques sur les entiers — Il existe quatre ***opérations élémentaires*** sur les bits : NOT, OR, AND et XOR. Leur résultat est présenté dans les tableaux ci-dessous.

a	NOT a
0	1
1	0

a	b	a OR b	a AND b	a XOR b
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Étant donné un k-plet $n = (a_k, \cdots, a_1)$ de bits ou, ce qui revient au même, un entier $n < 2^k$, on introduit alors également l'opération élémentaire de ***décalage à droite***, définie par

$$\mathrm{RS}(n) = (0, a_k \cdots, a_2).$$

On remarquera que l'entier correspondant à $\mathrm{RS}(n)$ est le quotient de la division euclidienne de n par 2 et a_1 est son reste. On définit de manière analogue le ***décalage à gauche*** en posant

$$\mathrm{LS}(n) = (a_{k-1}, \ldots a_1, 0).$$

On vérifie facilement que l'entier associé à $\mathrm{LS}(n)$ n'est autre que le reste de la division euclidienne de $2n$ par 2^k. En particulier, pour $a_k = 0$, ce qui se traduit par l'inégalité $n < 2^{k-1}$, on a l'identité $\mathrm{LS}(n) = 2n$.

Toutes les opérations usuelles entre entiers (comparaison, addition, soustraction, multiplication et division euclidienne) peuvent être effectuées par une suite d'opérations élémentaires sur leurs bits. Décrivons par exemple un algorithme permettant d'effectuer la somme de deux entiers naturels a et b à k bits correspondant respectivement aux k-plets $(a_k, \ldots, a_1)$, et $(b_k, \ldots, b_1)$:

$r \leftarrow 0$ (bit de retenue)
Pour i de 1 à k
    $c_i \leftarrow (a_i \,\mathrm{XOR}\, b_i) \,\mathrm{XOR}\, r$
    $r \leftarrow (a_i \,\mathrm{AND}\, b_i) \,\mathrm{OR} (a_i \,\mathrm{AND}\, r) \,\mathrm{OR} (b_i \,\mathrm{AND}\, r)$
$c_k \leftarrow r$

Cet algorithme nécessite $7k$ opérations élémentaires. Le résultat est le $(k+1)$-plet de bits $(c_{k+1}, \ldots, c_1)$ et l'entier qui lui est associé coïncide avec la somme de a et b. Le tableau ci-dessous indique le nombre (pas nécessairement minimal et arrondi par excès) d'opérations élémentaires nécessaires pour effectuer les opérations usuelles entre deux entiers naturels à k bits.

Comparaison	$2k$
Addition	$7k$
Soustraction	$8k$
Multiplication	$8k^2$
Division euclidienne	$13k^2$

1.2.4. Complexité d'un algorithme arithmétique sur les entiers — Comme il a été souligné précédemment, tous les algorithmes considérés dans ce cours se réduisent à une suite d'opérations arithmétique sur des entiers (comparaison, somme, produit, division euclidienne). On peut associer à un tel algorithme deux types de complexités : la ***complexité en espace***, qui mesure la mémoire (de l'ordinateur) nécessaire à son execution, et la ***complexité en temps***, qui est une évaluation de sa durée d'exécution. C'est cette deuxième notion qui nous intéresse principalement. Nous partirons du principe que toutes les opérations élémentaires sur les bits ont la même durée d'exécution et qu'elles seules contribuent de manière significative au temps d'exécution de l'algorithme (c'est clairement une simplification du problème, mais les résultats présentés restent néanmoins valables). En ce qui nous concerne, il est par conséquent naturel de définir la ***complexité*** de l'algorithme comme le nombre d'opérations élémentaires nécessaires pour qu'il aboutisse, ce nombre dépendant clairement des entiers sur lesquels on applique l'algorithme. Avec cette convention, le temps d'exécution est donc proportionnel à sa complexité. Cette constante de proportionnalité dépend de l'ordinateur utilisé et n'est pas d'un réel intérêt (tant que

l'on ne s'intéresse pas à l'implémentation effective de l'algorithme). Afin d'obtenir une expression qui ne tienne pas compte de ce facteur, il est utile d'introduire quelques notions de comparaison asymptotique. Considérons deux applications $f, g : \mathbb{N}_{>0} \to \mathbb{R}$. On dit que g est ***dominée*** par f, et on écrit $g = O(f)$, s'il existe une constante $c \in \mathbb{R}$ telle que

$$|g(n)| \leq c|f(n)|$$

pour tout n. D'après le paragraphe précédent, la complexité des opérations arithmétiques sur les entiers naturels peu être majorée par une fonction ne dépendant que de leur taille. Par exemple, nous avons vu que la complexité de l'algorithme d'addition de deux entiers de k bits est inférieure ou égale à $7k$. Avec la notation asymptotique introduite ci-dessus, la somme de deux entiers de k bits est donc de complexité $O(k)$. Le tableau suivant donne une expression de la complexité de quelques algorithmes de base sur deux entiers de k bits.

Algorithme	Complexité
Comparaison	$O(k)$
Addition/soustraction	$O(k)$
Multiplication/division euclidienne	$O(k^2)$

Remarque 1.2.8 — L'algorithme d'addition utilisé dans la pratique est très proche de celui décrit dans le paragraphe précédent. Concernant la multiplication (ou la division euclidienne), on a utilisé ici un algorithme qui n'est autre que l'adaptation en base 2 de la technique multiplication (ou de division) à la main, telle que l'on apprend à l'école. Ce dernier n'est pas nécessairement optimal d'un point de vue effectif, mais il l'est d'un point de vue asymptotique.

Il est souvent plus pratique d'estimer la complexité d'un algorithme en faisant intervenir les entiers considérés plutôt que leur taille. Tenant compte des relations $\ell_2(n) = \lfloor \log_2(n) \rfloor = O(\log(n))$, la somme de deux entiers naturels a et b est alors de complexité $O(\max\{\log(a), \log(b)\})$.

Dans le tableau ci-dessous, nous avons reporté l'estimation de la complexité en fonction des entiers naturels $b \leq a$ considérés.

Algorithme	Complexité
Comparaison	$O(\log(a))$
Comparaison, addition, soustraction	$O(\log(a))$
Multiplication, division euclidienne	$O(\log^2(a))$
Algorithme d'Euclide étendu	$O(\log^2(a))$

Remarque 1.2.9 — Concernant l'algorithme d'Euclide étendu, à chaque étape, on effectue une division euclidienne (afin de déterminer la suite des restes), deux multiplications et deux soustractions (pour le calcul des suites auxilières (u_n) et (v_n)). On

vérifie facilement que les entiers intervenant dans l'emsemble de l'algorithme sont inférieurs ou égaux à a, ce qui implique que chaque étape est de complexité $O(\log^2(a))$. Comme il a été noté précédemment (cf. la remarque 1.1.20), en utilisant la division euclidienne modifiée (qui est elle aussi de complexité $O(\log^2(a))$, l'algorithme aboutit après $\ell_2(a) = \lfloor \log_2(a) \rfloor + 1 = O(\log(a))$ étapes au plus (une estimation semblable est valable si l'on utilise la division euclidienne ordinaire). On en déduit que l'algorithme d'Euclide es de complexité $O(\log^3(a))$. Une étude plus fine (tenant compte du fait que la suite des restes est strictement décroissante) permet de montrer qu'il est en fait de complexité $O(\log^2(a))$.

Un algorithme sur des entiers naturels $a_1, \ldots, a_n$ est ***polynomial*** s'il existe des entiers naturels $d_1, \ldots, d_n$ tels que sa complexité soit égale à $O(\log^{d_1}(a_1) \cdots \log^{d_n}(a_n))$. On dit également que la complexité de l'algorithme est polynomiale. Tel est le cas par exemple pour les algorithmes d'addition, de soustraction, de multiplication, de division euclidienne ou de calcul de pgcd. Ce type d'algorithme est particulièrement adapté à des applications effectives, tout particulièrement en cryptographie, car en augmentant la taille des entiers considérés, la croissance de sa complexité reste modérée (polynomiale par rapport à leur taille, ce qui justifie d'ailleurs le nom).

1.2.5. Le problème de la factorisation — S'il est important en cryptographie de disposer d'algorithmes performants (i.e. de complexité polynomiale), de nombreux cryptosystèmes se basent par contre sur l'absence de tels algorithmes dans des problèmes particuliers, réputés difficiles d'un point de vue effectif. Tel est par exemple le cas pour la factorisation d'un entier : on ne dispose pas à ce jour d'algorithme rapide permettant de déterminer une factorisation non triviale d'un entier composé (i.e. non premier). Un chapitre de ce cours est dédié à cette question. Pour l'heure, remarquons qu'afin de factoriser un entier composé n, une première méthode naturelle (et incontournable) consiste à tester récursivement la divisibilité de n par tous les entiers inférieurs ou égaux à $\sqrt{n}$ (en effet, si n est composé, il possède un diviseur $1 < d \leq \sqrt{n}$), et ce, en effectuant à chaque étape une division euclidienne. On obtient alors un algorithme de complexité $O(\sqrt{n}\log^2(n))$. En d'autres termes, la croissance de sa complexité est bien plus que polynomiale par rapport à la taille de n (cette dernière étant égale à $\ell_2(n) = O(\log(n))$). Dans ce cas, elle est même ***exponentielle***. Nous verrons que même l'utilisation de techniques plus sophistiquées n'amène pas à une amélioration significative par rapport à la méthode décrite ici (la complexité reste essentiellement exponentielle).

1.3. Deux algorithmes performants

1.3.1. Exponentiation rapide — En cryptographie, il est souvent nécessaire de calculer des (grandes) puissances g^n d'un éléments g d'un groupe G (ou, plus généralement, d'un monoïde). Il existe une méthode naïve de calcul de g^n qui consiste

à effectuer $n-1$ multiplications, ce qui, dans la pratique, peut souvent se révéler prohibitif. Un procédé plus astucieux consiste à considérer l'écriture de n en base 2,

$$n = (a_\ell a_{\ell-1} \cdots a_0)_2 = a_0 + 2a_1 + \cdots + a_\ell 2^\ell,$$

où $\ell + 1 = \lfloor \log_2(n) \rfloor + 1$ est la taille de n. En posant $g_i = g^{2^i}$, on a alors l'identité

$$g^n = g_0^{a_0} \cdots g_\ell^{a_\ell}.$$

On remarquera que l'on a la relation $g_{i+1} = g_i^2$. On en déduit que ℓ multiplications (en fait, des élévations au carré) dans G suffisent pour déterminer les éléments $g_0, \ldots, g_\ell$. Les entiers a_i étant égaux à 0 ou 1, l'élément g^n est déterminé en effectuant au plus ℓ multiplication supplémentaires. En tout, on obtient un nombre de multiplications inférieur ou égal à 2ℓ. Cette méthode, appelée ***exponentiation rapide***, est donc performante et couramment utilisée dans la pratique. De nombreux cryptosystèmes (RSA, El-Gamal, Rabin,...) ou algorithmes (critères de primalité, méthodes de factorisation,...) seraient en effet absolument ineffectifs sans l'utilisation cette technique.

Exemple 1.3.1 — En utilisant la méthode d'exponentiation rapide, l'entier $3^{1000000}$ est déterminé en effectuant 27 multiplications dans $\mathbb{N}$. D'un point de vue pratique, sur un Macbook Pro, ce calcul nécessite environ $0,02$ secondes, contre 18 avec la méthode naïve (comportant 999999 multiplications).

1.3.2. Extraction de la racine carrée d'un entier — Nous terminons ce chapitre en décrivant un algorithme rapide pour déterminer la (partie entière de la) racine carrée d'un entier $n > 0$. En notant r le plus grand entier tel que $4^r \leq n$, ce qui se traduit par l'identité $r = \lfloor \log_2(n)/2 \rfloor$, considérons la suite (x_i) définie par $x_0 = 0$ et

$$x_{i+1} = \begin{cases} x_i + 2^{r-i} \text{ si } (x_i + 2^{r-i})^2 \leq n, \\ \\ x_i \text{ sinon.} \end{cases}$$

Proposition 1.3.2 — *L'élément x_{r+1} est la partie entière de $\sqrt{n}$.*

Démonstration — Montrons par récurrence que pour tout $i \in \{1, \ldots, r+1\}$, x_i est un entier naturel vérifiant les inégalités

$$x_i^2 \leq n < (x_i + 2^{r-i+1})^2.$$

Pour $i = 1$, on a l'identité $x_1 = 2^r$, d'où les relations

$$x_1^2 = 4^r \leq n < 4^{r+1} = (x_1 + 2^r)^2.$$

Soit donc $i \in \{1, \ldots, n\}$ et supposons que x_i est un entier naturel vérifiant les inégalités ci-dessus. Pour $(x_i + 2^{r-i})^2 \leq n$, on obtient $x_{i+1} = x_i + 2^{r-i} \in \mathbb{N}$, d'où les relations

$$x_{i+1}^2 \leq n < (x_i + 2^{r-i+1})^2 = (x_{i+1} + 2^{r-i})^2$$

Pour $(x_n + 2^{r-i})^2 > n$, on obtient $x_{i+1} = x_i \in \mathbb{N}$ et les relations

$$x_{i+1}^2 = x_i^2 \leq n < (x_i + 2^{r-i})^2 = (x_{i+1} + 2^{r-i})^2.$$

Finalement, pour $i = r + 1$, on obtient les inégalités

$$x_{r+1}^2 \leq n < (x_{r+1} + 1)^2,$$

d'où le résultat. □

Corollaire 1.3.3 — *L'algorithme d'extraction de la racine carrée d'un entier naturel n est de complexité $O(\log^3(n))$.*

Démonstration — La détermination de l'entier r nécessitant $2r$ décalages vers la gauche et r comparaisons d'entiers inférieurs ou égaux à n, elle est de complexité $O(r\log(n) + 2r) = O(\log^2(n))$. À chaque pas de l'algorithme, on effectue au plus une élévation au carré, une comparaison, un décalage à droite et une somme, le tout avec des entiers inférieurs ou égaux à n^2, ce qui amène à une complexité globale de $O((r + 1)(\log^2(n)) = O(\log^3(n))$. □