

## Mémento complexité

**Rappel.** Soit  $g$  une fonction strictement positive ; on écrit

- $f = O(g)$  s'il existe  $C > 0$  et  $x_0$  tel que  $f(x) \leq C \cdot g(x)$  pour tout  $x > x_0$ .
- $f = \tilde{O}(g)$  s'il existe  $C > 0$  et  $x_0$  tel que  $f(x) \leq g(x) \cdot (\log g(x))^C$  pour tout  $x > x_0$ .

### 1. TRI

Le tri de  $n$  éléments s'effectue en utilisant  $O(n \log n)$  comparaisons (tri fusion).

### 2. EXPONENTIATION BINAIRE

Le calcul de  $x^n$  s'effectue en utilisant  $O(\log n)$  multiplications (écriture de  $n = \sum_{i \leq t} n_i 2^i$  en base 2 et calcul des  $x^{2^i}$  en utilisant  $t$  carrés successifs).

### 3. DANS $\mathbb{Z}$

La *taille* (binaire) d'un entier  $a$  est le nombre de bits<sup>1</sup> utilisé pour l'écrire, en d'autres termes  $t(a) := \lfloor \log_2(|a| + 1) \rfloor + 1$ . En général,  $a > 1$  tend vers l'infini et on utilise simplement  $\log_2(a)$ . On suppose que tous les opérandes sont de taille  $\leq n$  :

Opération	Naïf	Rapide
$a + b$	$O(n)$	$O(n)$
$a \times b$	$O(n^2)$	$M_{\mathbb{Z}}(n) = \tilde{O}(n)$
$a = bq + r$	$O(n^2)$	$O(M_{\mathbb{Z}}(n))$
Relation de Bézout	$O(n^2)$	$O(M_{\mathbb{Z}}(n) \cdot \log n)$
Restes chinois	$O(n^2)$	$O(M_{\mathbb{Z}}(n) \cdot \log n)$

Pour tous ces algorithmes, on compte le nombre d'opérations élémentaires sur les bits, supposé proportionnel au temps d'exécution (complexité en *temps*). La complexité en *espace* est de  $\tilde{O}(n)$  bits.

- Multiplication :  $M_{\mathbb{Z}}(n)$  est par définition le temps d'une multiplication dans  $\mathbb{Z}$  pour deux opérandes de taille  $\leq n$ . L'algorithme rapide (Harley et van der Hoeven, 2019), en temps  $O(n \log n)$ , utilise la Transformée de Fourier Discrète.

Moins rapide mais beaucoup plus simple : l'algorithme de Karatsuba utilise  $O(n^{\log_2 3})$  opérations.

**Remarque.** Pour des opérantes de tailles différentes,  $s(a) \leq n$ ,  $s(b) \leq m$ , l'algorithme naïf (quadratique) utilise  $O((n+1)(m+1))$  opérations.

<sup>1</sup>Il n'y a pas d'inconvénient à compter les chiffres en base  $\beta \geq 2$ , avec  $t(a) := \lfloor \log_{\beta}(|a| + 1) \rfloor$ . Pas vraiment d'intérêt théorique non plus.

- Division euclidienne : l'entrée est  $(a, b)$ ,  $b \neq 0$ , et la sortie  $(q, r)$  avec  $0 \leq r < |b|$ . L'algorithme rapide résout l'équation  $b - a/x = 0$  en utilisant l'itération de Newton

$$x_{n+1} = x_n - x_n(x_n b - a).$$

(Calcul effectué en doublant la précision à chaque étape ; soit dans un modèle flottant, soit modulo  $2^k$ .) On pose  $q = \lfloor x \rfloor$ , puis  $r = a - bq$ . L'énoncé de complexité suppose que  $M_{\mathbb{Z}}(n)$  vérifie des propriétés du type  $M(n)/n \geq M(m)/m$  pour  $n \geq m$ , et  $M(mn) \leq m^2 M(n)$ . Qui sont vraies aussi bien pour l'algorithme naïf que pour Karatsuba ou Harley - van der Hoeven.

**Remarque.** Si  $t(a) \leq n$ ,  $t(b) \leq m$ ,  $n \geq m$ , l'algorithme naïf (quadratique) utilise  $O((n - m + 1)(m + 1))$  opérations élémentaires.

- Bézout : l'entrée est  $(a, b)$  et la sortie consiste en  $\text{pgcd}(a, b)$  et deux entiers  $u, v$  tels que  $au + bv = \text{gcd}(a, b)$ .
- Restes chinois : l'entrée consiste en  $N$  congruences  $x \equiv a_k \pmod{b_k}$  où les  $b_k$  sont 2 à 2 premiers entre eux, et la sortie attendue est une solution de ce système de congruences. On suppose  $t(a_k) \leq t(b_k)$  and  $\sum_k t(b_k) \leq n$ .

#### 4. DANS $\mathbb{Z}/N\mathbb{Z}$

On choisit un représentant canonique dans chaque classe de congruence : les entiers de  $[0, N - 1]$ , pour représenter les éléments de  $\mathbb{Z}/N\mathbb{Z}$ . La taille des opérandes est  $\leq n := t(N)$ .

L'addition est implantée comme une addition dans  $\mathbb{Z}$ , suivie d'une (possible) soustraction : temps  $O(n)$ . La multiplication est une multiplication in  $\mathbb{Z}$ , suivie par une division euclidienne par  $N$  : temps  $O(n^2)$  ou  $O(M_{\mathbb{Z}}(n))$ . L'inversion utilise une relation de Bézout : temps  $O(n^2)$  ou  $O(M_{\mathbb{Z}}(n) \cdot \log n)$ .

#### 5. DANS $K[X]$ , $K$ UN CORPS

La *taille* (algébrique) d'un polynôme  $h \in K[X]$  est son nombre de coefficients :  $t(h) := \deg h + 1 \leq n$ .

Pour la complexité, on choisit de compter les opérations dans  $K$  (complexité algébrique) :  $+$ ,  $-$ ,  $\times$ ,  $/$  dans  $K$ . Pour estimer plus réalistement le temps de calcul, on peut multiplier par le coût d'une opération élémentaire dans  $K$  quand celui-ci est borné uniformément (c'est le cas si  $K$  est un corps fini) ; sinon il faut tenir compte de la taille des coefficients.

Soient  $f, g$  deux polynômes dans  $K[X]$ , de tailles  $\leq n$ .

Opération	Naïf	Rapide
$f + g$	$O(n)$ [+]	$O(n)$ [+]
$f \times g$	$O(n^2)$ [+ , $\times$ ]	$M_{K[X]}(n) = \tilde{O}(n)$
$f = gh + r$	$O(n^2)$	$O(M_{K[X]}(n))$
Bézout	$O(n^2)$	$O(M_{K[X]}(n) \cdot \log n)$
Restes chinois	$O(n^2)$	$O(M_{K[X]}(n) \cdot \log n)$

### 6. DANS $K[X]/(T)$ :

Deux cas particuliers importants : définitions des corps finis non premiers  $\mathbb{F}_q$  comme extension algébrique de  $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$  et des extensions de  $\mathbb{Q}$  (nombres algébriques). De façon analogue à  $\mathbb{Z}/N\mathbb{Z}$  nous travaillons avec des polynômes de taille  $\leq t(T)$ , avec les coûts ci-dessus. L'inversion utilise de nouveau une relation de Bézout.

### 7. DANS $M_{n \times n}(K)$ :

On se place de nouveau dans le modèle de complexité algébrique : on compte les opérations dans le corps  $K$ . Pour  $A \in M_{n \times n}(K)$ , sa *taille* est  $t(A) = n^2$  ; pour  $v \in K^n$  sa taille est  $t(v) := n$ . On note  $A, B$  deux matrices dans  $M_n(K)$ .

Opérations	Naïf	Rapide
$A + B$	$O(n^2)$	$O(n^2)$
$Av, v \in K^n$	$O(n^2)$	$O(n^2)$
$A \times B$	$O(n^3)$	$O(n^\omega), \omega = 2.3729$
$A^{-1}$	$O(n^3)$	$O(n^\omega)$
$PA = LU$	$O(n^3)$	$O(n^\omega)$
Char $A$	$O(n^3)$	$O(n^\omega)$

La factorisation  $LU$  (pivot de Gauss) permet de résoudre la plupart des problèmes d'algèbre linéaire sur  $K$  : noyau, image, rang, déterminant, inverse, système linéaire ( $LU$  + résolution de systèmes triangulaires en  $O(n^2)$  opérations supplémentaires dans  $K$ ).

La constante  $\omega \leq 3$  qui apparaît ci-dessus est appelé *exposant de la multiplication matricielle*. La valeur utilisée en pratique (pour des matrices physiquement représentables par les ordinateurs actuels) est  $\omega = \log_2 7 \approx 2.8$  (Strassen).

**Algèbre linéaire « Boîte Noire ».** Dans ce modèle, les coûts sont exprimés en terme du nombre d'évaluations  $v \mapsto A \cdot v$  pour un "opérateur boîte noire"  $A$ . (Un opérateur linéaire dont on ne suppose pas qu'il est donné par une matrice. On sait simplement calculer l'image de tout vecteur  $v \in K^n$ .) En général, une multiplication matrice-vecteur pour  $A \in M_{n \times n}(K)$  et  $v \in K^n$  utilise  $O(n^2)$  opérations dans  $K$  mais la plupart des matrices rencontrées en pratique sont « structurées » et permettent une évaluation plus économique :

matrices par bandes, plus généralement matrices creuses (comme dans l'algorithme de Dixon de factorisation dans  $\mathbb{Z}$ ); matrice de Sylvester définissant le résultant; matrice de Berlekamp (pour factoriser les polynômes sur les corps finis); matrices de van der Monde (comme les matrices de FFT), etc.

Dans ce modèle il existe des algorithmes *probabilistes* (Lanczos, Wiedemann) qui résolvent efficacement des problèmes d'algèbre linéaire comme le calcul du polynôme caractéristique ou du rang de  $A$  (avec faible probabilité d'erreur dans les 2 cas), ou trouve *une* solution du système linéaire  $Ax = b$ . Ils utilisent en moyenne  $O(n)$  évaluations  $A \cdot v$ , plus  $O(n^2)$  opérations dans  $K$ . On ne gagne rien sur une matrice dense générale ( $A \cdot v$  utilise  $O(n^2)$  opérations dans  $K$ ); mais le gain est significatif si  $A \cdot v$  se calcule plus rapidement.