1. Opérations basiques sur les entiers et les polynomes - Notions de complexité

1.1. La notation $O(.)$

Elle sert à donner un ordre de grandeur par opposition à une valeur précise. On s'en sert pour les tailles d'objets et les temps de calcul.

Il n'est pas forcément pertinent de dire qu'un algorithme manipulant des objets de taille binaire n utilise $3\,n^4+27$ opérations machine puisque cette quantité eut par exemple varier en fonction de la machine utilisée. Par exemple multiplier 2 entiers machine de 64 bits sur une machine 64 bits utilise 1 opération machine alors qu'il peut en utiliser beaucoup plus sur une machine 16 bits.

Par contre , le fait que $3\,n^4+27$ se comporte comme n^4 lorsque n grandit est une information éventuellement pertinente. On dit alors que l'algorithme se comporte en $O(n^4)$.

Pour formaliser un peu , voici les principales règles :

On considère des fonctions $f, g: \mathbb{N} \to \mathbb{R}$

- Si $c \in \mathbb{R}$ est une constante, $c\,O(f) = O(f)$
- $O(f) + O(g) = O(f+g) = O(\max(f, g))$
- $O(f)\,O(g) = O(f\,g) = f\,O(g)$

Attention :

$O(f)+O(f)=O(f)$ mais $\underbrace{O(f)+\ldots+O(f)}_{n \text{ fois}} = n\,O(f) = O(n\,f)$ (n n'est pas une constante mais, par abus de langage, une fonction de $\mathbb{N} \to \mathbb{R}$)

1.2. Représentations

Entiers multi-précision

$$e = \text{sign}(e)\,(a_0 + a_1\,2^{64} + \ldots. + a_\tau\,2^{\tau 64}), a_i \in [0 \ldots 2^{64}-1]$$

(Tableau d'entiers machine 64 bits)

Taille binaire $\boldsymbol{\tau}(e) \in O(\lceil \log_2(|e|) \rceil) \sim O(\tau)$

Avec la notation $O(.)$ la taille de ce tableau est aussi la longeur binaire ou la longueur dans une base arbitrairement choisie :

$$e = \text{sign}(e)\,(a_0 + a_1\,2 + ... + a_\tau\,2^{\tau'}), a_i \in \{0,1\}, \tau' = \log_2(|e|)$$

Rationnels

$$r = \frac{e_n}{e_d}$$

(Couple d'entiers machine)

Taille binaire $\tau(r) \in O(\tau(e_n) + \tau(e_d))$ mais aussi $\tau(r) \in O(\max(\tau(e_n), \tau(e_d)))$

Polynôme

$$P(X) = a_0 + a_1\,X + ... + a_d\,X^d, a_i \in \mathbb{Z},\ \tau(a_i) \leqslant \tau$$

(Tableau de coefficients)

Stricto sensu $\tau(P) = \sum_{i=0}^{d} \tau(a_i) \leqslant (d+1)\ \tau \in O(d\,\tau)$. Le plus souvent, on distingue le degré et la taille des coefficients.

1.3. Additions/soustractions

Addition/Soustraction d'entiers

$e_a = \text{sign}(e_a)(a_0 + a_1\,2^{64} + ... + a_\tau\,2^{\tau_a 64}), a_i \in [0...2^{64} - 1]$
$e_b = \text{sign}(e_b)(b_0 + b_1\,2^{64} + ... + b_\tau\,2^{\tau_b 64}), b_i \in [0...2^{64} - 1]$

$|e_c| = |e_a| + |e_b| = \sum_{i=0}^{\tau_c} (|a_i| + |b_i|) 2^{i64}$ avec

- $\tau_c = \max(\tau_a, \tau_b)$
- $|a_i| + |b_i| \in [0...2^{64}]$

$|e_c| = \sum_{i=0}^{\tau_c+1} |c_i| 2^{i64}$

$\tau(e_c) \leqslant \max(\tau(e_a), \tau(e_b)) + 1$

Complexité binaire du calcul : au plus $2\ \max(\tau_a,\ \tau_b) + 1 \leqslant 2\ \tau + 1 \in O(\tau)$ opérations machine pour une taille qui augmente de 1.

Addition/Soustraction de polynomes

P_a de degré $d_a \leqslant d$ et P_b de degré $d_b \leqslant d$

Complexité arithmétique du calcul : au plus $3 \max(d_a, d_b) + 1 \leqslant 3\,d + 1 \in O(d)$ opérations (additions/soustractions) entre les coefficients.

Si les coefficients des 2 polynômes sont au plus de taille τ, alors le coût est de l'ordre de $O(d\tau)$ opérations binaires.

Quelques pièges usuels :

$\underbrace{e+....+e}_{n}$ est de taille $O(\log(n)\,\tau)=\tilde{O}(\tau)$

Comparer 2 entiers de taille τ coute $O(\tau)$ opérations binaires.

1.4. Multiplication naïve

Entiers

$e_1\,e_2=(a_0+a_1\,2^{64}+....+a_{\tau_1}\,2^{\tau_1 64})\,(b_0+b_1\,2^{64}+....+b_{\tau_2}\,2^{\tau_2 64})=a_0\,b_0+...+a_{\tau_1}b_{\tau_2}2^{(\tau_1+\tau_2)64}$

Résultat de taille au moins $\tau_1+\tau_2$ mais attention $|a_i\,b_j|\in[0,...,2^{128}-1]$.

On écrit $|a_i\,b_i|=\underline{c_i}+\overline{c_i}2^{64}$ de sorte que $e_1\,e_2=\sum_{i=0}^{\tau_1+\tau_2}\underline{c_i}\,2^{i64}+\sum_{i=1}^{\tau_1+\tau_2+1}\overline{c_i}2^{i64}$,en remarquant que le résultat de cette addition sera de longueur au plus $\tau_1+\tau_2+1$ (pas de retenue sur le dernier mot machine).

Résultat de taille au plus $\tau_1+\tau_2+1$

Nombre d'opérations arithmétiques : $\tau_1\,\tau_2$ multiplications machine puis au plus $2\,(\tau_1+\tau_2+1)$ additions machine soit :

$O(\tau_1\tau_2)$ opérations machine pour un résultat de taille $O(\tau_1+\tau_2)$

On retiendra : $O(\tau^2)$ operations machine pour une taille qui double.

Polynômes

P_1 de degré $d_1\leqslant d$ et P_2 de degré $d_2\leqslant d$ avec des coefficients de taille au plus τ

Il en coute $O(d_2d_1)$ ou $O(d^2)$ opérations arithmétiques pour un résultat de degré au plus d_1+d_2 ou $2\,d$.

En écrivant $P_1\,P_2=(\sum_{i=0}^{d_1}a_iX^i)\,(\sum_{j=0}^{d_2}b_jX^j)=\sum_{k=0}^{d_1+d_2}c_k\,X^k$ on voit que chaque c_k est au plus la somme de d_1+d_2+1 produits $a_i\,b_j$ et donc $\tau(c_k)\leqslant 2\log_2(2\,d+1)\tau$

Ainsi les opérations arithmétiques sont des opérations sur les entiers de taille d'ordre $O(\tau\log(d))$ ou $\tilde{O}(\tau)$ le cout total est donc de l'ordre de $\tilde{O}(d^2\tau)$.

1.5. Intermède : l'exponentiation rapide

2. Multiplication rapide

Définition 1. $\mathbb{K}$ *un anneau,* $n \in \mathbb{N}^\star$ $w \in \mathbb{K}$ *est une racine n-ieme de l'unité si* $w^n = 1$. *On dit qu'elle est primitive d'ordre n si de plus* $w^{\frac{n}{t}} - 1$ *n'est pas un diviseur de zéro pour tout diviseur premier t de n.*

Lemme 2. $w \in \mathbb{K}$ *une racine n-ieme primitive de l'unité et* $l \in \mathbb{N}, 1 < l < n$. *Alors*

- $w^l - 1$ *n'est pas diviseur de zéro*
- $\sum_{0 \leqslant j < n} w^{lj} = 0$

Définition 3. $f = \sum_{i=0}^{n-1} f_i X^i \in \mathbb{K}[X]$ *et* $\vec{f} = [f_0, ..., f_{n-1}] \in \mathbb{K}^n$

Discrete Fourier Transform :

$$\begin{array}{rccccc} \mathrm{DFT}_w\colon & \mathbb{K}[X] & \approx & \mathbb{K}^n & \to & \mathbb{K}^n \\ & f & \leftrightarrows & \vec{f} & \longmapsto & (f(1), f(w), f(w^2), ..., f(w^{n-1})) \end{array}$$

Convolution de polynômes :

$h = f \star_n g = \sum_{0 \leqslant l < n} h_l\, X^l$, *avec* $h_l = \sum_{j+k \equiv l \bmod n} f_j\, g_k = \sum_{0 \leqslant j < n} f_j\, g_{l-j}$ *pour* $0 \leqslant l < n$

Lemme 4. $f \star_n g = f\, g \bmod X^n - 1$.
En particulier, si f, g *sont 2 polynômes de degrés au plus* $n-1$ *,alors* $f \star_{2n} g = f\, g$.

Lemme 5.
$\mathrm{DFT}_w(f \star_n g) = \mathrm{DFT}_w(f) \times \mathrm{DFT}_w(g)$ *ou* $\times$ *est le produit terme à terme.*

Démonstration. En divisant $f \star_n g$ par $X^n - 1$ on obtient $q \in \mathbb{K}[X]$ tel que $f \star g = f\, g + q\,(X^n - 1)$. Ainsi $(f \star g)(w^j) = f\,(w^j)\; g(w^j) + q(w^j)\,(w^{jn} - 1) = f\,(w^j)\; g(w^j)$. □

$$V_w = \mathrm{VDM}(1, w, ..., w^{n-1}) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{pmatrix} = (w^{jk})_{0 \leqslant j,k < n} \in \mathbb{K}^n \times \mathbb{K}^n$$

Lorsque lon multiplie V_w par $\vec{f}$ on évalue f en les w^i. Autrement dit : $\mathrm{DFT}_w(\vec{f}) = V_w \vec{f}$.

2.1. Cas d'un anneau contenant des racines de l'unité de tous ordres.

Exemple : $\mathbb{C}$.

THÉORÈME 6. *$\mathbb{K}$ anneau commutatif unitaire, $n \in \mathbb{N}_{\geqslant 1}$ et $w \in \mathbb{K}$ racine primitive n-ieme de l'unité. Alors w^{-1} est racine primitive n-ième de l'unité et $V_w\, V_{w^{-1}} = n\,\mathrm{Id}_n$.*

Ainsi, $n^{-1}V_{w^{-1}} = (V_w)^{-1}$. Lorsque l'on calcule $\frac{1}{n}V_{w^{-1}}\,[\alpha_0, ..., \alpha_{n-1}]^T$ on détermine les coefficients de l'unique polynôme h de degré au plus $n-1$ tel que $h(w^i) = \alpha_i$.

Principe :

- On choisit k tel que $\deg(f\,g) < 2^k = n$ et on considère w racine n ieme de l'unité dans $\mathbb{K}$.
- On calcule $\mathrm{DFT}_w(\vec{f})$ puis $\mathrm{DFT}_w(\vec{g})$ puis leur produit produit terme à terme $\mathrm{DFT}_w(f \star_n g) = \mathrm{DFT}_w(f) \times \mathrm{DFT}_w(g) = \mathrm{DFT}_w(f\,g \bmod X^n - 1) = \mathrm{DFT}_w(f\,g)$
- Enfin $f\,g = \frac{1}{n}\mathrm{DFT}_{w^{-1}}(\mathrm{DFT}_w(f\,g))$

Le produit terme à terme coutant $O(n)$ opérations arithmétiques, l'effort est porté sur les calculs de DFT.

Réaliser l'opération $\mathrm{DFT}_w(\vec{f}) = V_w\vec{f}$ par ce produit Matrice/vecteur n'a aucun intérêt puisque ceci consomme $O(n^2)$ opérations arithmétiques.

On décompose $f = f_1\, X^{2^{k-1}} + f_0$ et on pose $r_0 = f_0 + f_1$ et $r_1 = f_0 - f_1$, ce qui permet d'obtenir $f = q_0\,(X^{2^{k-1}} - 1) + r_0 = q_1\,(X^{2^{k-1}} + 1) + r_1$ et on remarque alors que $f(w^{2l}) = r_0(w^{2l})$ et $f(w^{2l+1}) = r_1(w^{2l})$ pour tout $l = 0...2^{k-1}$.

Ainsi, évaluer f en les $w^i, i = 0...2^k$ revient à évaluer r_0 en les puissances paires de w et r_1 en les puissances impaires de w.

Or w^2 est une racine primitive 2^{k-1} de l'unité, calculer $\mathrm{DFT}_w(f)$ revient donc à calculer $\mathrm{DFT}_{w^2}(r_0)$ et $\mathrm{DFT}_{w^2}(r_1')$ avec $r_1'(X) = r_1(w\,X)$.

Ceci donne un algorithme dont le nombre d'opérations arithmétiques dans $\mathbb{K}$ est défini par $T(2^k) = 2\,T(2^{k-1}) + O(2^k)$ soit $T(n) \in O(n\log(n))$ ou encore $\tilde{O}(n)$.

2.2. Cas presque général (Schönhage et Strassen).

La seule supposition sur $\mathbb{K}$ est que 2 y soit inversible. Lorsque $\mathbb{K}$ ne contient pas de racines de l'unité (ou de racines de l'unité d'ordre suffisant), ou simplement si on en connait pas, l'idée est d'introduire des « racines virtuelles de l'unité » ce qui revient en fait à travailler symboliquement dans divers anneaux de coefficients de la forme $\mathbb{D} = \frac{\mathbb{K}[y]}{y^m+1}$.

LEMME 7. *Si 2 est inversible dans $\mathbb{K}$ et si m est une puissance de 2, alors $\omega = X$ mod $(X^m + 1)$ est une racine $2\,m$-ème principale de l'unité dans $\mathbb{D} = \frac{\mathbb{K}[X]}{X^m+1}$.*

Démonstration. Comme $w^m = -1$, w est bien une racine $2\,m$-ème de l'unité. Soit t un diviseur strict de $2m$. Puisque m est une puissance de 2, t divise m, et donc $w^t - 1$ divise $w^n - 1 = -2$ qui est supposé inversible, ce qui montre que $w^t - 1$ n'est pas diviseur de zéro, ce qui implique que w est primitive. □

On se donne 2 polynômes $F, G \in \mathbb{K}[X]$ de degrés au plus $n = 2^k$ et on définit $m = 2^{\lfloor k/2 \rfloor}$, $t = n \,/\, m$ ainsi que les polynômes F_0, ..., F_{t-1}, G_0, ..., G_{t-1} tels que $F'(X,\, Y) = \sum_{j=0}^{t-1} F_j(X)\; Y^j$ et $G'(X,\, Y) = \sum_{j=0}^{t-1} G_j(X)\; Y^j$ de sorte à ce que $F(X) = F'(X, X^m)$ et $G(X) = G'(X, X^m)$.

$F'\,G'$ est alors un polynôme de degré strictement inférieur à $2\,t \leqslant 4\,m$ en Y et ses coefficients sont des polynômes de degrés en X strictement inférieurs à $2\,m$.

On remarque que X est une racine $4m$-ième de l'unité dans $\mathbb{D} = \frac{\mathbb{K}[X]}{X^{2m}+1}$, ce qui fait que l'on peut appliquer l'algorithme de la section précédente pour calculer le produit $H' = F'\,G'$ dans $\mathbb{D}[Y]$ en $\tilde{O}(m)$ opérations arithmétiques dans $\mathbb{D}$. Il suffit ensuite de remarquer que $H(X) = H'(X,\, X^m) = F'(X,\, X^m)\; G'(X,\, X^m) = F(X)\,G(X)$ pour conclure que ce produit est le produit désiré.

D'après la secrtion précédente, il en coutera $\tilde{O}(t)$ opérations arithmétiques dans $\mathbb{D}$.

Les opérations arithmétiques dans $\mathbb{D}$ sont essentiellement des multiplications de polynômes de degrés $O(m)$ à coefficients dans $\mathbb{K}$ combinées avec des additions de polynômes à coefficients dans $\mathbb{K}$ de degrés $O(m)$ pour réduire modulo $X^{2m}+1$.

On montre ainsi par récurrence sur m que le calcul du produit $F'\,G'$ dans $\mathbb{D}[Y]$ est en $\tilde{O}(t)\;\tilde{O}(m) = \tilde{O}(n)$ dans $\mathbb{K}$ et comme les coefficients de ce produit sont de degrés strictement inférieur à $2\,m$, on obtient alors le produit $F'\,G'$ dans $\mathbb{K}[X, Y]$.

Il ne reste plus qu'à calculer $F'G'(X,\, X^m)$ en remplaçant Y pas X^m, ce qui revient à additionner terme à terme t vecteurs de longeur $O(m)$ pour un cout total de $\tilde{O}(n)$ opérations dans $\mathbb{K}$.

Cas des entiers

$f = \sum_{i=0}^{n-1} f_i\, 2^{i64}$ et $g = \sum_{i=0}^{n-1} g_i\, 2^{i64}$ et on suppose $h = f\,g = \sum_{i=0}^{n-1} h_i\, 2^{i64}$.

On forme $F(X) = \sum_{i=0}^{n-1} f_i\; X^i$ et $G(X) = \sum_{i=0}^{n-1} g_i\; X^i$ que l'on multiplie pour obtenir $H = FG = \sum_{i=0}^{n-1} H_i\, X^i$. On a alors $f\,g = F(2^{64})\,G(2^{64}) = H(2^{64})$.

Le souci est que l'on se ramène au produit de 2 polynômes de $\mathbb{Z}[X]$ et 2 n'est pas inversible dans $\mathbb{Z}$, on ne peut donc pas appliquer l'algorithme précédent pour calculer .

On remarque que si $0 \leqslant f_i,\, g_i < 2^{64}$ alors $0 \leqslant H_i < 2^{128+\log_2(n)}$ et on se souvient que si $m = 2^k$, alors 2 est une racine m-ieme de l'unité dans $\frac{\mathbb{Z}}{(2^m+1)\mathbb{Z}}$. Le produit FG peut donc s'effectuer dans $\frac{\mathbb{Z}}{(2^m+1)\mathbb{Z}}[X]$ en choisissant $m > 128 + \log_2(n)$.

Ainsi, multiplier 2 entiers de taille τ coutera $\tilde{O}(\tau)$ opérations dans $\frac{\mathbb{Z}}{(2^m+1)\mathbb{Z}}$ ce qui donne, par récurrence sur m un cout total en $\tilde{O}(\tau)$ opérations binaires.

Cas des polynômes à coefficients entiers

$F = \sum_{i=0}^{n-1} f_i X^i$ et $G = \sum_{i=0}^{n-1} g_i X^i$ et on suppose $H = FG = \sum_{i=0}^{n-1} h_i X^i$.

On pose $L \in O(\tau + \log(n))$ de telle sorte que $|h_i| < 2^{L-1}$

On forme $f = \sum_{i=0}^{n-1} f_i 2^{iL}$ et $g = \sum_{i=0}^{n-1} g_i 2^{iL}$

On calcule $fg = \sum_{i=0}^{n-1} h_i 2^{iL}$ puis on en déduit $FG = \sum_{i=0}^{n-1} h_i X^i$

Attention aux signes !

2.3. Rationnels

Remarque : rationnels $\frac{e_1}{e_1'}$ de taille $\mu_1 = \max(\tau_1, \tau_1')$ additions et multiplications naïves se comportent comme des multiplications d'entiers (coût et taille).

3. DivisionS

Rappel : Les anneaux Euclidiens sont des anneaux commutatifs intègres dans lesquels il existe une division Euclidienne.

Ils sont munis d'une application $\nu : \mathbb{E} \setminus \{0\} \to \mathbb{N}$ appelée stathme permettant de comparer 2 éléments non nuls.

$\forall P_1, P_2 \in \mathbb{E}, \nu(P_2) \leqslant \nu(P_1), \exists!(R, Q), P_1 = Q\,P_2 + R, \nu(R) < \nu(P_2)$.

Exemples : $\mathbb{Z}$, $||$, $\mathbb{Q}[X]$, deg(.) et plus généralement $\mathbb{F}[X]$, deg(.) ou $\mathbb{F}$ est un corps.
Par contre, $\mathbb{Q}[X][Y]$ n'est pas Euclidien, $\mathbb{Z}[X]$ non plus.

Exercice : dans $\mathbb{Q}[X]$ l'opération de division Euclidienne (naïve) prend $O(d_2\, d_1)$ opérations dans $\mathbb{Q}$.

3.1. Pseudo-division

Dans $\mathbb{Z}[X]$ on a un souci dès lors que le diviseur n'est pas unitaire. Mais dans $\mathbb{Z}[X]$ et en général dans $\mathbb{K}[X]$ ou $\mathbb{K}$ est un anneau factoriel, on peut « tricher » en introduisant la pseudo-division. Au lieu de calculer $P_1 = Q\,P_2 + R$ on va calculer $a\,P_1 = Q\,P_2 + R$.

Exercice : On pose $a = \mathrm{Lc}(P_2)^{\mathrm{degree}(P_1)-\mathrm{degree}(P_2)}$ et on peut alors remarquer que la division Euclidienne de $a\,P_1$ par P_2 produit $Q, R \in \mathbb{K}[X]$ si $P_1, P_2 \in \mathbb{K}[X]$.

Exercice : quel est le coût d'une pseudo division en opérations arithmétiques, en opérations binaires dans le cas de $\mathbb{Z}[X]$.