Algorithmique et langage C: monoïde, corps finis et Euclide étendu .

Michaël Quisquater (Maître de Conférences, UVSQ)

Calcul de puissance rapide : tentative

Considérons un G un ensemble muni d'une oopération binaire $\cdot$ associative, et soit un naturel $d \in \mathbb{N}$ et $x \in G$.

Calcul de $x^d = x \cdot x \cdot x \cdots x$

Méthode élémentaire : $d - 1$ multiplications

Remarque : en cryptographie, d est très grand et donc ce n'est pas praticable.

Tentatives d'amélioration

Calcul de 2^6.

Par la méthode élémentaire : 5 multiplications.

Idée : $6 = 2 \cdot 3$. Par conséquent, $2^6 = (2^3)^2$.

$\rightarrow$ 2 multiplications + 1 multiplication (carré) : 3 opérations au total.

Remarque : Notons que cette méthode fonctionne car 6 est factorisable en le produit de 2 et de 3.

Amélioration : Première méthode (intuition)

Calcul de 2^5.

Méthode de base : 4 multiplications et la méthode précédente ne fonctionne plus.

Idée : décomposition binaire de l'exposant : $5 = 2^2 + 0 \cdot 2^1 + 2^0$.

$$2^5 = 2^{2^2} \cdot 2^{2^0}.$$

Il suffit de calculer $2^{2^0} = 2$, $2^{2^1} = 2^2$ et $2^{2^2} = (2^2)^2$.

$\rightarrow$ 2 multiplications

Ensuite multiplier 2^{2^0} et 2^{2^2}

$\rightarrow$ 3 multiplications au total au lieu de 4.

Amélioration : Première méthode (formalisation)

bits faibles → forts

Décomposer l'exposant en binaire :

$$d = d_k 2^k + d_{k-1} 2^{k-1} + \cdots + d_1 2 + d_0$$

A chaque étape, multiplication éventuelle par une puissance x^{2^i}

"*square-and-multiply*"

$$x^d = (x^{2^k})^{d_k} \cdot (x^{2^{k-1}})^{d_{k-1}} \cdots (x^{2^1})^{d_1} \cdot (x^{2^0})^{d_0}$$

Remarque : la séquence des puissances x^{2^i} peut être calculée efficacement car $x^{2^i} = (x^{2^{i-1}})^2$.

"Square-and-multiply" : bits faibles → bits forts

Algorithme 1 : "Square-and-multiply"

Données : $x \in G$, $d \in \mathbb{N}$.
Résultat : x^d
$d = \sum_{i=0}^{k} d_i \cdot 2^i$ (avec $d_k = 1$), $temp := 1$. $puiss = x$
pour $i = 0, \ldots, k-1$ **faire**
  si $d_i = 1$ **alors**
    $temp := temp \cdot puiss$
  fin
  $puiss := puiss^2$
fin
retourner $temp \cdot puiss$

Premirère méthode (exemple)

Calculons $[7 + 11\mathbb{Z}]^5$ ou de façon équivalente $7^5 \bmod 11$.

Initialisation : $5 = 2^2 + 2^0$ ($d_2 = 1$, $d_1 = 0$ $d_0 = 1$). $k = 2$
$x = 7$, $temp = 1$ et $puiss = 7$.

itération 0 : $d_0 = 1$ $temp = temp \cdot puiss = 1 \cdot 7 \bmod 11 = 7$,
$puiss = puiss^2 \bmod 11 = 7^2 \bmod 11 = 5$

itération 1 : $d_1 = 0$ $-$,
$puiss = puiss^2 \bmod 11 = 5^2 \bmod 11 = 3$

return : $temp \cdot puiss = 7 \cdot 3 \bmod 11 = 10$,

Conclusion : $[7 + 11\mathbb{Z}]^5 = [10 + 11\mathbb{Z}]$

Amélioration : Seconde méthode (intuition)

Calcul de 11^{13}.

Méthode élementaire : 12 multiplications.

Idée : Séquence de divisions Euclidiennes (division par 2)

$$13 = 2 \cdot 6 + 1, 6 = 2 \cdot 3 + 0, 3 = 2 \cdot 1 + 1$$

On calcule $11^3 = (11)^2 \cdot 11^1$, $11^6 = (11^3)^2$ et $11^{13} = (11^6)^2 \cdot 11$.

$\rightarrow$ 5 multiplications.

Seconde méthode (formalisation)

bits forts → faibles

Décomposer l'exposant en binaire :

$$d = d_k 2^k + d_{k-1} 2^{k-1} + \cdots + d_1 2 + d_0$$

$$x^d = ((((x^{d_k})^2 \cdot x^{d_{k-1}})^2 \cdot x^{d_{k-2}} \ldots)^2 \cdot x^{d_1})^2 \cdot x^{d_0}$$

A chaque étape, élévation au carré et éventuellement multiplication par x.

"*square-and-multiply*"

"Square-and-multiply" : bits forts → bits faibles

Algorithme 2 : "Square-and-multiply"

Données : $x \in G$, $d \in \mathbb{N}$.
Résultat : x^d
$d = \sum_{i=0}^{k} d_i \cdot 2^i$ (avec $d_k = 1$), $temp := x$.
pour $i = k-1, \ldots, 0$ **faire**
  $temp := temp^2$
  si $d_i = 1$ **alors**
    $temp := temp \cdot x$
  fin
fin
retourner *temp*

Seconde méthode (exemple)

Calculons $[7 + 11\mathbb{Z}]^5$ ou de façon équivalente $7^5 \bmod 11$.

Initialisation : $5 = 2^2 + 2^0$ ($d_2 = 1$, $d_1 = 0$ $d_0 = 1$). $k = 2$.
$x = 7$ et $temp = 7$.

itération 1 : $temp = temp^2 \bmod 11 = 7^2 \bmod 11 = 5$
$d_1 = 0$ $-$,

itération 0 : $temp = temp^2 \bmod 11 = 5^2 \bmod 11 = 3$
$d_0 = 1$ $temp = temp \cdot x = 3 \cdot 7 \bmod 11 = 10$,

Conclusion : $[7 + 11\mathbb{Z}]^5 = [10 + 11\mathbb{Z}]$

Complexité des méthodes d'exponentiation rapide

Les deux méthodes nécessitent de réaliser au plus $2 \cdot log_2\ d$ ($log_2\ d$ est la taille de d) opérations dans G.

Conclusion : Si l'opération de G est polynomiale (en la taille des éléments de G), cela signifie que l'exponentiation est une opération polynomiale.

Génération de générateur d'un groupe fini

Soit G un groupe fini tel que $|\ G\ | = \prod_{i \in I} p_i^{\alpha_i}$ avec p_i premiers distincts et $\alpha_i \in \mathbb{N} \setminus \{0\}$.

Observation :

Soit $x \in G$ et e l'élément neutre de G. On sait que l'ordre de x divise l'ordre de G, donc l'ordre de x est de la forme $\prod_{j \in J} p_j^{\beta_j}$ avec $J \subseteq I$ et $\beta_j \leq \alpha_j$ pour tout $j \in J$.

Par conséquent, si $x^{\frac{|G|}{p_i}} \neq e$ pour un certain $i \in I$, cela implique que l'ordre de x ne divise pas $\frac{|G|}{p_i}$ et donc est de la forme $p_i^{\alpha_i} \cdot \prod_{j \in J \setminus \{i\}} p_j^{\beta_j}$ avec $J \subseteq I$ et $\beta_j \leq \alpha_j$ pour tout $j \in J$. Par conséquent, l'ordre de x est un multiple de $p_i^{\alpha_i}$.

Génération de générateur d'un groupe fini (suite)

Si $x^{\frac{|G|}{p_i}} \neq e$ pour tout $i \in I$, cela signifie que que l'ordre de x est un multiple du $ppcm\{p_i^{\alpha_i} \mid i \in I\}$.

Or $ppcm\{p_i^{\alpha_i} \mid i \in I\} = \prod_{i \in I} p_i^{\alpha_i}$ car les p_i's sont distincts et donc relativement premiers.

Finalement, l'ordre de x divise l'ordre du groupe c-à-d $\prod_{i \in I} p_i^{\alpha_i}$.

On peut conclure que l'ordre de x est $\prod_{i \in I} p_i^{\alpha_i}$.

Génération de générateur d'un groupe fini

Algorithme 3 : Génération de générateur d'un groupe fini

Données : G cyclique d'ordre $\prod_{i\in I} p_i^{\alpha_i}$ premiers distincts et $\alpha_i \in \mathbb{N}\setminus\{0\}$.
Résultat : un générateur x de G

```
pour x ∈ G faire
    pour i ∈ I faire
        temp = x^(|G|/p_i)
        si temp = e alors
            break
        fin
    fin
    retourner x
fin
```


Astuce de Montgomery

Données : $x_1, x_2, \cdots, x_n$ des éléments inversibles d'un monoïde commutatif G.
Résultat : $x_1^{-1}, x_2^{-1}, \cdots, x_n^{-1}$.

Idée : Soit $a, b \in G$. Alors,

$$a^{-1} = (a\cdot b)^{-1}\cdot b\,,$$

$$b^{-1} = (a\cdot b)^{-1}\cdot a\,,$$

Observation : 3 multiplications et 1 inversion permettent de calculer l'inverse de deux éléments.

Astuce de Montgomery (suite)

Astuce de Montgomery

Generalisation de cette observation à n éléments
$\Rightarrow$ l'inversion de n éléments nécessite $3(n-1)$ multiplications et 1 inversion.

Idée : Soit $x_1, \cdots, x_n$ des éléments inversibles de G.

$$x_n^{-1} = (x_1 \cdots x_n)^{-1} \cdot (x_1 \cdots x_{n-1}) .$$

$\Rightarrow$ Appliquer ce principe de façon récursive.

Algorithme 4 : Astuce de Montgomery

Données : $x_1, x_2, \cdots, x_n$ des éléments inversibles d'un monoïde commutatif G.
Résultat : $x_1^{-1}, x_2^{-1}, \cdots, x_n^{-1}$.

$Prod[1] = x_1$
pour $i = 2 \cdots n$ **faire**
  $Prod[i] = Prod[i-1] \cdot x_i$
fin
$temp = (Prod[n])^{-1}$
pour $i = n \cdots 2$ **faire**
  $Inv_x[i] = temp \cdot Prod[i-1]$
  $temp = temp \cdot x_i$
fin
$Inv_x[1] = temp$
retourner Inv_x

Introduction

- Réaliser les opérations dans les anneaux finis demandent beaucoup de calculs.
- Lorsque l'anneau fini est de petite taille, il est évidemment possible de précalculer les tables de Cayley pour l'addition et la multiplication.
  - Gain de temps substantiel pour les calculs futurs.
  - Demande beaucoup de mémoire : $2 * q^2$ "cases" pour un corps fini $\mathbb{F}_q$.
- Dans le cas des corps finis, il est possible de faire beaucoup mieux !

Rappels sur les corps finis

- Un corps fini à q^n élément est une extension de degré n d'un corps fini $\mathbb{F}_q$ et peut être construit via un quotient

$$\mathbb{F}_q[X]/(P(X))$$

  où $(P(X))$ est l'idéal engendré par un polynôme irréductible (de $\mathbb{F}_q[X]$) de degré n.
- Le groupe multiplicatif d'un corps fini $\mathbb{F}_q$ est cyclique et possède $\phi(q-1)$ générateurs (ϕ l'indicatrice d'Euler).

Rappels sur les corps finis (suite)

- Un polynôme primitif $P(X)$ d'un corps fini $\mathbb{F}_{q^n}$, extension de $\mathbb{F}_q$, est un polynôme minimal (appartenant à $\mathbb{F}_q[X]$) d'un générateur de $\mathbb{F}_{q^n}^*$.
- Un polynôme primitif est irréductible et est souvent utilisé pour la construction de corps finis car $[X + (P(X))]$ est générateur du groupe cyclique du corps fini.
- Le nombre de polynômes primitifs d'un corps $\mathbb{F}_{q^n}$, extension de $\mathbb{F}_q$, est $\phi(q^n - 1)/n$.

Tabulation "polynomiale/exponentielle" des corps finis

Considérons le corps fini $\mathbb{F}_{q^n}$.

- Les classes peuvent être représentées par un polynôme de $\mathbb{F}_q[X]$ de degré strictement inférieur à n.
- On parlera de représentation "polynomiale.

$\Rightarrow$ représentation pour l'addition (simplement additionner les polynômes de $\mathbb{F}_q[X]$) et pour le calcul de l'opposé.

Tabulation "polynomiale/exponentielle" des corps finis

- Si $\underline{\alpha}$ est un générateur de $\mathbb{F}_{q^n}^*$, tout élément de $\mathbb{F}_{q^n}^*$ est de la forme $\underline{\alpha}^i$ avec $i \in \mathbb{Z}_{q^n-1}$.
- Chaque élément de $\mathbb{F}_{q^n}^*$ peut être représenté par une classe de $\mathbb{Z}_{q^n-1}$.
- On parlera de représentation "exponentielle".

$\Rightarrow$ représentation pour la multiplication (simplement additionner les exposants appartenant à $\mathbb{Z}_{q^n-1}$), l'inverse et l'extraction de racines.

Tabulation "polynomiale/exoponentielle" des corps finis : addition et multiplication

- L'idée de la tabulation "polynomiale/exponentielle" consiste à effectuer les additions en représentation "polynomiale" et les multiplications en représentation "polaire".
- Dénotons par $\underline{\alpha}$ un générateur de $\mathbb{F}_{q^n}^*$.
- Définissons $Exp_{\underline{\alpha}} : \mathbb{Z}_{q^n-1} \longrightarrow \mathbb{F}_{q^n}^* : i \mapsto \underline{\alpha}^i$.
- $Log_{\underline{\alpha}}$ est l'application réciproque de $Exp_{\underline{\alpha}}$.

Remarque : Dans la suite, les représentants minimaux seront utilisés pour identifier classes. Une classe de représentant $Q(X)$ sera notée $\underline{Q(X)}$.

Tabulation "polynomiale/exponentielle" des corps finis : addition et multiplication

Soit les classes $\underline{S(X)}$ et $\underline{T(X)}$ de $\mathbb{F}_{q^n}$.

- L'addition de ces deux classes consiste à simplement additionner $S(X)$ et $T(X)$ pour obtenir $\underline{S(X) + T(X)}$.
- La multiplication s'obtient via :

$$\underline{S(X)} \cdot \underline{T(X)} = \begin{cases} Exp_{\underline{\alpha}}(Log_{\underline{\alpha}}(\underline{S(X)}) + Log_{\underline{\alpha}}(\underline{T(X)})) & \text{si } \underline{S(X)}, \underline{T(X)} \in \mathbb{F}_{q^n}^* \\ 0 & \text{sinon} \end{cases}$$

Remarques :

- L'addition de classes se réduit à l'addition de vecteur de taille n à coefficients dans $\mathbb{F}_q$. Il s'agit d'un XOR si $q = 2$.
- La multiplication de classes se réduit à une addition dans $\mathbb{Z}_{q^n-1}$ et à des évaluations des applications exponentielle et logarithme.

Tabulation "polynomiale/exponentielle" des corps finis : opposé, inverse et racine $n^{ième}$

- L'opposé d'une classe $\underline{S(X)}$ de $\mathbb{F}_{q^n}$ est donnée par :

$$-\underline{S(X)} = \underline{-S(X)}\,.$$

- L'inverse d'une classe $\underline{S(X)}$ de $\mathbb{F}_{q^n}^*$ est donnée par :

$$\underline{S(X)} = Exp_{\underline{\alpha}}(-Log_{\underline{\alpha}}(\underline{S(X)}))$$

- La racine $n^{ième}$ d'une classe $\underline{S(X)}$ de $\mathbb{F}_{q^n}^*$ (quand elle existe) est donnée par :

$$\underline{S(X)} = Exp_{\underline{\alpha}}(n^{-1} \cdot Log_{\underline{\alpha}}(\underline{S(X)}))$$

où n^{-1} est calculé dans $\mathbb{Z}_{q^n-1}$.

Avantages et inconvients de la tabulation "polynomiale/exponentielle"

Avantages :

- Les éléments sont représentés sous forme polynomiale (intéressant si on doit utiliser la structure d'espace vectoriel d'un corps fini)
- L'addition d'éléments est très rapide (surtout en caractéristique 2).

Inconvénients :

- La multiplication est assez lente (de multiples appels aux applications exponentielles et logarithmes)

Introduction

- Une idée alternative pourrait être de considérer la représentation exponentielle comme celle de référence.
- Dans ce cas, c'est la multiplication qui devient très rapide ; il suffit d'additionner les exposants dans $\mathbb{Z}_{q^n-1}$.
- L'addition devient l'opération "complexe". En effet, l'exposant de la somme de deux éléments dont les exposants sont m et n est :

$$Log_{\underline{\alpha}}(Exp_{\underline{\alpha}}(m) + Exp_{\underline{\alpha}}(n)) .$$

$\Rightarrow$ il possible de simplifier cette formule, c'est l'objet du logarithme de Zech (ou de Jacobi) !

Tabulation : Logarithme de Zech dans les corps finis

Soit $m, n \in \mathbb{Z}_{q^n-1}$ (ou quelque chose comme cela...)

Observons que

$$Exp_{\underline{\alpha}}(m) + Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(m)(1 + Exp_{\underline{\alpha}}(n-m)) .$$

Par conséquent,

$$Log_{\underline{\alpha}}(Exp_{\underline{\alpha}}(m) + Exp_{\underline{\alpha}}(n)) = m + Log_{\underline{\alpha}}(1 + Exp_{\underline{\alpha}}(n-m)) .$$

Définissons le logarithme de Zech par :

$$Z_{\underline{\alpha}}(k) = Log_{\underline{\alpha}}(1 + Exp_{\underline{\alpha}}(k))$$

Dans ce cas, on a :

$$Log_{\underline{\alpha}}(Exp_{\underline{\alpha}}(m) + Exp_{\underline{\alpha}}(n)) = m + Z_{\underline{\alpha}}(n-m) .$$

Tabulation : Logarithme de Zech dans les corps finis

Notons que le logarithme de Zech est défini sur $\mathbb{Z}_{q^n-1} \setminus \{e\}$ où e est tel que $Exp_{\underline{\alpha}}(e) = -1$.
Aussi, cette méthode ne permet pas de considérer la somme dont un ou plusieurs éléments sont nuls. Finalement l'image de ce logarithme est $\mathbb{Z}_{q^n-1} \setminus \{0\}$.
L'astuce consiste à compléter $\mathbb{Z}_{q^n-1}$ du symbole $-\infty$ avec les conventions suivantes :

- $Exp_{\underline{\alpha}}(-\infty) = 0$
- $k + (-\infty) = -\infty$
- $Z_{\underline{\alpha}}(-\infty) = 0$
- $Z_{\underline{\alpha}}(e) = -\infty$ avec e tel que $Exp_{\underline{\alpha}}(e) = -1$

Tabulation : Logarithme de Zech dans les corps finis

Avec ces conventions :

1. $Exp_{\underline{\alpha}}(m) + Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(m + Z_{\underline{\alpha}}(n-m))$
2. $-Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(e) \cdot Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(n+e)$
3. $Exp_{\underline{\alpha}}(m) - Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(m + Z_{\underline{\alpha}}(e+n-m))$
4. $Exp_{\underline{\alpha}}(m) \cdot Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(m+n)$
5. $Exp_{\underline{\alpha}}(m)^{-1} = Exp_{\underline{\alpha}}(-m)$
6. $Exp_{\underline{\alpha}}(m)/Exp_{\underline{\alpha}}(n) = Exp_{\underline{\alpha}}(m) \cdot (Exp_{\underline{\alpha}}(n))^{-1} = Exp_{\underline{\alpha}}(m-n)$

Remarque : Les formules (1) et (3) ne sont pas définies quand $m = -\infty \Rightarrow$ cas à traiter indépendamment.

Calcul de pgcd : Première méthode

Première méthode : conséquence du théorème fondamental de l'arithmétique :

Corollaire

Considérons les naturels non-nuls a et b et leurs factorisations $a = \prod_{i \in I} p_i^{\alpha_i}$ *et* $b = \prod_{j \in J} p_j^{\beta_j}$ *avec* $I, J \subset \mathbb{N}_0$ *et avec* $\alpha_i, \beta_j \in \mathbb{N}_0$ *pour* $i \in I$ *et* $j \in J$*. Alors, un plus grand commun diviseur de ces nombres est donné par la formule :*

$$pgcd(a,b) = \prod_{i \in I \cap J} p_i^{\min(\alpha_i, \beta_i)}.$$

De plus, si a est un naturel non-nul et b est nul, alors $pgcd(a,0) = a$*. Notons que par convention* $\prod_{i \in \emptyset} a_i = 1$*.*

Calcul de pgcd : Première méthode (exemple)

Exemple : Considérons les nombres $15 = 3 \cdot 5$ et $18 = 2 \cdot 3^2$. Nous avons

$$15 = \prod_{i \in I} p_i^{\alpha_i} \text{ avec } I = \{2,3\}, p_2 = 3, p_3 = 5 \text{ et } \alpha_2 = 1, \alpha_3 = 1\,,$$

et

$$18 = \prod_{i \in J} p_i^{\beta_i} \text{ avec } J = \{1,2\}, p_1 = 2, p_2 = 3 \text{ et } \beta_1 = 1, \beta_2 = 2\,.$$

Par conséquent, $I \cap J = \{2\}$. Il s'ensuit

$$pgcd(15,18) = \prod_{i \in I \cap J} p_i^{\min(\alpha_i,\beta_i)} = p_2^{min(1,2)} = 3^{min(1,2)} = 3\,.$$

Calcul de pgcd : Deuxième méthode

La première méthode nécessite la factorisation des nombres
$\rightarrow$ difficile !

Une autre méthode existe et est appelée "algorithme d'Euclide". Elle est basée sur le résultat suivant :

Théorème

Considérons les entiers $a, b, c \in \mathbb{Z}$ avec a et b non-nuls simultanément. Alors,

$$pgcd(a,b) = pgcd(a + b \cdot c, b)\,.$$

Calcul de pgcd : Deuxième méthode (suite)

Preuve. Soit $d_1 = pgcd(a, b)$ et $d_2 = pgcd(a + b \cdot c, b)$, $c \in \mathbb{Z}$
Montrons que $d_1 \mid d_2$. Par définition du pgcd, d_1 divise a et b. Par conséquent, d_1 divise $a + b \cdot c$ et b. Par la définition du pgcd, nous déduisons que $d_1 \mid d_2$ ou encore

$$d_2 = s \cdot d_1 \text{ pour } s \in \mathbb{Z}. \quad (1)$$

Montrons que $d_2 \mid d_1$. Appliquons le point précédent aux nombres $a = a + b \cdot c$ et $b = b$ et $c = -c$. Nous avons que $pgcd(a + b \cdot c, b) \mid pgcd((a + b \cdot c) + b \cdot (-c), b) = pgcd(a, b)$ ou encore

$$d_1 = s' \cdot d_2 \text{ pour } s' \in \mathbb{Z}. \quad (2)$$

(2) et (1) $\rightarrow s \cdot s' = 1$. Par conséquent, $s = s' = 1$ ou $s = s' = -1$. Le résultat suit. □

Calcul de pgcd : Deuxième méthode (algorithme d'Euclide)

$pgcd(126, 35)$?

Observons que $126 = 35 \cdot 3 + 21$ (on divise 126 par 35). Donc,

$$pgcd(126, 35) = pgcd(35 \cdot 3 + 21, 35) = pgcd(35, 21)$$

De même, $35 = 21 \cdot 1 + 14$. Donc,

$$pgcd(35, 21) = pgcd(21 \cdot 1 + 14, 21) = pgcd(21, 14)$$

Aussi, $21 = 14 \cdot 1 + 7$. Donc,

$$pgcd(21, 14) = pgcd(14 \cdot 1 + 7, 14) = pgcd(14, 7)$$

Finalement, $14 = 7 \cdot 2 + 0$. Donc,

$$pgcd(14, 7) = pgcd(7 \cdot 2 + 0, 7) = pgcd(7, 0) = 7$$

Calcul de pgcd : Deuxième méthode (suite)

Conclusions : Pour calculer le plus grand commun diviseur de deux entiers r_0 et r_1 (non-nuls simultanément), il suffit d'effectuer la séquence des divisions Euclidiennes :

$$\begin{aligned} r_0 &= r_1 \cdot q_1 + r_2 \\ r_1 &= r_2 \cdot q_2 + r_3 \\ r_2 &= r_3 \cdot q_3 + r_4 \\ &\dots \\ r_{n-2} &= r_{n-1} \cdot q_{n-1} + r_n \\ r_{n-1} &= r_n \cdot q_n + 0 \end{aligned}$$

Le dernier reste non-nul est le $pgcd(r_0, r_1)$
Remarque : la séquence s'arrêtera toujours car les restes r_i's sont strictement décroissants.

Calcul de pgcd : Deuxième méthode (algorithme d'Euclide)

Algorithme 5 : Algorithme d'Euclide : Calcul de pgcd

Données : $a, b \in \mathbb{Z}$ non-nuls simultanément.
Résultat : $pgcd(a, b)$
$r_0 =\mid a \mid$, $r_1 =\mid b \mid$, $k = 1$.
tant que $r_k \neq 0$ **faire**
  r_{k+1} := le reste de la division de r_{k-1} par r_k
  $k = k + 1$
fin
retourner $pgcd(a, b) = r_{k-1}$.

Calcul de pgcd : Algorithme d'Euclide (exemple) : bis

Exemple : $pgcd(126, 35)$

$$r_0 = 126,\ r_1 = 35,\ r_2 = 21,\ r_3 = 14,\ r_4 = 7 \text{ et } r_5 = 0.$$

Le pgcd est donc 7.

Remarque : si on prend le plus petit reste positif lors de la division Euclidienne, le pgcd obtenu sera toujours le pgcd positif.

Théorème de Bezout

Théorème

(Théorème de Bezout) Soit deux entiers a et b non simultanément nuls. Alors, il existe $x, y \in \mathbb{Z}$ tel que

$$a \cdot x + b \cdot y = pgcd(a, b)\,.$$

Les nombres x et y sont appelés les coefficients de Bezout.

Théorème de Bezout (intuition de la preuve)

Reprenons notre exemple : soit $a = 126$ et $b = 35$.

On cherche $x, y \in \mathbb{Z}$ tels que $126x + 35y = pgcd(126, 25) = 7$

Considérons la séquence de divisions Euclidiennes :

$$126 = 35 \cdot 3 + 21 \qquad 21 = 126 - 35 \cdot 3$$
$$35 = 21 \cdot 1 + 14 \qquad 14 = 35 - 21 \cdot 1$$
$$21 = 14 \cdot 1 + 7 \qquad 7 = 21 - 14 \cdot 1$$
$$14 = 7 \cdot 2 + 0 \qquad 0 = 14 - 7 \cdot 2$$

But : Exprimer 7 en fonction de 126 et 35.

Méthode : exprimer successivement 126, 35, 21,14 et 7 en fonction de 126 et 35

Théorème de Bezout (intuition de la preuve)

$$r_0 = 126 = 1 \cdot 126 + 0 \cdot 35$$
$$r_1 = 35 = 0 \cdot 126 + 1 \cdot 35$$
$$r_2 = 21 = 126 - 35 \cdot 3$$
$$r_3 = 14 = 35 - 21 \cdot 1$$
$$r_4 = 7 = 21 - 14 \cdot 1$$
$$r_5 = 0 = 14 - 7 \cdot 2$$

$$r_0 = x_0 \cdot 126 + y_0 \cdot 35$$
$$r_1 = x_1 \cdot 126 + y_1 \cdot 35$$
$$r_2 = (x_0 \cdot 126 + y_0 \cdot 35) - q_1 \cdot (x_1 \cdot 126 + y_1 \cdot 35)$$
$$r_3 =$$
$$r_4 =$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= (x_0 - q_1 x_1) \cdot 126 + (y_0 - q_1 y_1) \cdot 35 \\
r_3 &= \\
r_4 &=
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= \\
r_4 &=
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= (x_1 \cdot 126 + y_1 \cdot 35) - q_2 \cdot (x_2 \cdot 126 + y_2 \cdot 35) \\
r_4 &=
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= (x_1 - q_2 x_2) \cdot 126 + (y_1 - q_2 y_2) \cdot 35 \\
r_4 &=
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= x_3 \cdot 126 + y_3 \cdot 35 \\
r_4 &=
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= x_3 \cdot 126 + y_3 \cdot 35 \\
r_4 &= (x_2 \cdot 126 + y_2 \cdot 35) - q_3 \cdot (x_3 \cdot 126 + y_3 \cdot 35)
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= x_3 \cdot 126 + y_3 \cdot 35 \\
r_4 &= (x_2 - q_3 x_3) \cdot 126 + (y_2 - q_3 y_3) \cdot 35
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

$$
\begin{aligned}
r_0 &= 126 = 1 \cdot 126 + 0 \cdot 35 \\
r_1 &= 35 = 0 \cdot 126 + 1 \cdot 35 \\
r_2 &= 21 = 126 - 35 \cdot 3 \\
r_3 &= 14 = 35 - 21 \cdot 1 \\
r_4 &= 7 = 21 - 14 \cdot 1 \\
r_5 &= 0 = 14 - 7 \cdot 2
\end{aligned}
$$

$$
\begin{aligned}
r_0 &= x_0 \cdot 126 + y_0 \cdot 35 \\
r_1 &= x_1 \cdot 126 + y_1 \cdot 35 \\
r_2 &= x_2 \cdot 126 + y_2 \cdot 35 \\
r_3 &= x_3 \cdot 126 + y_3 \cdot 35 \\
r_4 &= x_4 \cdot 126 + y_4 \cdot 35
\end{aligned}
$$

Théorème de Bezout (intuition de la preuve)

Conclusion :

- $x_0 = 1$, $y_0 = 0$, $x_1 = 0$ et $y_1 = 1$
- $x_{k+1} = x_{k-1} - q_k \cdot x_k$
- $y_{k+1} = y_{k-1} - q_k \cdot y_k$
- Les x_i et y_i correspondants au dernier reste non-nuls sont les coefficients cherchés.

Remarque : Le principe de la preuve consiste à montrer que $a \cdot x_i + b \cdot y_i = r_i$ pour tout i.

Algorithme d'Euclide étendu : Calcul des coefficients de Bezout.

Algorithme 6 : Algorithme d'Euclide Etendu

Données : $a, b \in \mathbb{Z}$ non-nuls simultanément.
Résultat : $pgcd(a, b)$ et $x, y \in \mathbb{Z}$ tels que $ax + by = pgcd(a, b)$
$r_0 =\mid a \mid$, $r_1 =\mid b \mid$, $x_0 = 1$, $x_1 = 0$, $y_0 = 0$, $y_1 = 1$, $k = 1$.

tant que $r_k \neq 0$ **faire**
  r_{k+1} := reste de la division de r_{k-1} par r_k;
  q_k := quotient de la division de r_{k-1} par r_k;
  $x_{k+1} = -q_k \cdot x_k + x_{k-1}$;
  $y_{k+1} = -q_k \cdot y_k + y_{k-1}$;
  $k = k + 1$
fin
retourner $pgcd(x, y) = r_{k-1}$, $x = x_{k-1}$, $y = y_{k-1}$.

Algorithme d'Euclide étendu (exemple)

- Exemple : $a = 126$ et $b = 35$.

k	0	1	2	3	4	5
r_k	126	35	21	14	7	0
q_k	-	3	1	1	2	-
x_k	1	0	1	-1	2	-
y_k	0	1	-3	4	-7	-

On a donc que le $pgcd(126, 35) = 7$ et $x = 2$, $y = -7$.

Par conséquent, $126 \cdot 2 - 7 \cdot 35 = 7$.

Remarque : Poser $r_0 = max(|\ a\ |, |\ b\ |)$ et $r_0 = min(|\ a\ |, |\ b\ |)$ permet de ne pas perdre une étape.

Algorithme d'Euclide étendu (simplification)

Notons finalement que la suite définie par

$$x'_i = x'_{i-2} + q_{i-1} \cdot x'_{i-1} \text{ (avec } x'_0 = 1 \text{ et } x'_1 = 0)$$
$$\text{resp. } y'_i = y'_{i-2} + q_{i-1} \cdot y'_{i-1} \text{ (avec } y'_0 = 0 \text{ et } y'_1 = 1)$$

est liée à la suite

$$x_i = x_{i-2} - q_{i-1} \cdot x_{i-1} (\text{resp. } y_i = y_{i-2} - q_{i-1} \cdot y_{i-1})$$

par la relation

$$x_i = (-1)^i \cdot x'_i \text{ (resp. } y_i = (-1)^{i+1} \cdot y'_i).$$

Algorithme d'Euclide étendu (simplification)

Manier les suites x'_i et y'_i est plus confortable → pas d'erreur de signe (ou utilisation d'*unsigned int*).

En pratique, on peut donc utiliser les suites x'_i et y'_i et on fera le changement de signe adéquat à la fin.

En particulier,

$$pgcd(r_0, r_1) = r_n = r_0 \cdot (-1)^n \cdot x'_n + r_1 \cdot (-1)^{n+1} \cdot y'_n .$$

Remarque : Le principe de la preuve consiste à montrer que $a \cdot (-1)^i \cdot x'_i + b \cdot (-1)^{i+1} \cdot y'_i = r_i$ pour tout i.

Algorithme d'Euclide étendu bis : Calcul des coefficients de Bezout.

Algorithme 7 : Algorithme d'Euclide Etendu bis

Données : $a, b \in \mathbb{Z}$ non-nuls simultanément.
Résultat : $pgcd(a, b)$ et $x, y \in \mathbb{Z}$ tels que $ax + by = pgcd(a, b)$
$r_0 =\mid a \mid$, $r_1 =\mid b \mid$, $x_0 = 1$, $x_1 = 0$, $y_0 = 0$, $y_1 = 1$, $k = 1$.

tant que $r_k \neq 0$ **faire**
  r_{k+1} := reste de la division de r_{k-1} par r_k;
  q_k := quotient de la division de r_{k-1} par r_k;
  $x_{k+1} = q_k \cdot x_k + x_{k-1}$;
  $y_{k+1} = q_k \cdot y_k + y_{k-1}$;
  $k = k + 1$
fin
retourner $pgcd(x, y) = r_{k-1}$, $x = (-1)^{k-1} x_{k-1}$, $y = (-1)^k y_{k-1}$.

Algorithme d'Euclide étendu bis (exemple)

- Exemple : $a = 126$ et $b = 35$.

k	0	1	2	3	4	5
r_k	126	35	21	14	7	0
q_k	-	3	1	1	2	-
x_k	1	0	1	1	2	-
y_k	0	1	3	4	7	-

On a donc que le $pgcd(126, 35) = 7$,

$$x = (-1)^{5-1} \cdot 2 = 2 \text{ et } y = (-1)^5 \cdot 7 = -7\,.$$

Par conséquent, $126 \cdot 2 - 7 \cdot 35 = 7$

Algorithme Euclide étendu

- La complexité de l'algorithme d'Euclide étendu est $\mathcal{O}(m \cdot n)$ opérations sur les mots (exemple 32 bits), où m et n sont les tailles (en bits) de a et b respectivement.
- Il existe de nombreuses variantes de l'algorithme d'Euclide étendu (binaire, Lehmer, demi-pgcd etc). Certains de ces algorithmes sont plus efficaces que l'algorithme d'Euclide étendu.
- Beaucoup de ces algorithmes se généralisent aux domaines Euclidien généraux. En particulier, il existe pour chacun de ces algorithmes un version pour les anneaux des polynômes à coefficients dans un corps.