Chapitre 15

Algorithmique des entiers

1 Multiprécision des entiers :

On utilise un calculateur où l'entier le plus long qu'on puisse représenter est ω ($-\omega \leq$ tout entier machine $\leq \omega$).
Les algorithmes de travail avec les grands nombres (GN) sont :

- lecture/écriture des GN
- conversion d'un entier en GN
- comparaison des GN
- arithmétique (addition, soustraction, multiplication, division euclidienne) des GN.

Soit un autre entier caractéristique nommé Base, noté B
Soit $a \in \mathbb{N}$, $a = a_{n-1}B^{n-1} + \cdots + a_1 B + a_0$, $\quad 0 \leq a_i < B \quad i = 0, \ldots, n-1$
<u>Définition</u> : On appelle longueur de a le nombre $l(a) = n$, a peut être rangé dans un tableau à n éléments.

2 Addition de deux entiers :

Soient

- $a = (a_{n-1}, a_{n-2}, \cdots, a_0)$
- $b = (b_{n-1}, b_{n-2}, \cdots, b_0)$
- $a + b = c = (c_{n-1}, c_{n-2}, \cdots, c_0)$

On calcule d'abord $a_0 + b_0$, puis $a_i + b_i + r$ (r la retenue)
$r \leq 1$, $a_i + b_i + r \leq (B-1) + (B-1) + 1 = 2B - 1 \leq \omega$
Par exemple calcul en base binaire (si $a < 2^m$ on a $l(a) = \lfloor \log_2 a + 1 \rfloor$)
Algorithme :

```
procédure (a, b, c)
(* a = (a_{n-1}, ···, a_0), b = (b_{n-1}, ···, b_0), c résultat, c=a+b *)
debut
r := 0, i := 0
l := max(l(a), l(b))
tant que i ≤ l-1 faire
t := a_i + b_i + r
c_i := t mod B
r := t div B
i := i + 1

Si r ≠ 0 alors c_l := r
fin
```

Coût opérations : 3 opérations par passage (2 additions + 1 addition) soit au total $3l$ opérations élémentaires, soit $O(l)$

3 Soustraction de deux entiers :

```
   procédure (a, b, c)
(* c = a-b, a > b *)
debut
r := 0, i := 0, l := l(a)
tant que i ≤ l-1 faire
t:= a_i-b_i-r
Si t < 0 alors c_i := B+t; r := 1; i := i + 1
sinon c_i := t; r := 0; i := i + 1
fin
```

4 Multiplication de deux entiers :

1) $l(a) = n$, $l(b) = 1$ $(b < B)$
On fait des opérations $a_i \times b$ + retenue

```
procédure multisimp(a, b, c)
(* b < B c = a*b a = (a_{n-1}, ···, a_0) *)
debut
r := 0, i := 0
tant que i ≤ l(a)-1 faire
t := a_i × b + r
c_i := t mod B
r := t div B
i := i + 1

Si r ≠ 0 alors c_l(a) := r
fin
```

2) $a = (a_{m-1}, \cdots, a_0)$
$b = (b_{n-1}, \cdots, b_0)$
on considère $\beta_k = b_k \cdots b_o$, $\beta_{n-1} = B$
$a\beta_k = (ab_k)B^k + a\beta_{k-1}$
Les k derniers chiffres de $a\beta_k$ et de $a\beta_{k-1}$ sont les mêmes, le nombre de chiffres de $a\beta_k \leq m + k + 1$.
Pour connaître $a\beta_k$ connaissant $a\beta_{k-1}$, il suffit de connaître les $m + 1$ chiffres.
Notons $C_i^{(k)}$ les chiffres de $a\beta_k$ $i = 0, \cdots, m + k$
On les connaît pour $i = 0, \cdots, k - 1$ car $C_i^{(k)} = C_i^{(k-1)}$

```
procédure pro(k)
(* Calcul des (m+1) premiers éléments de aβ_k *)
debut
Pour i := 0 jusqu'à m-1 faire
t := a_i × b_k + C_{k+i}^{(k-1)} + r
C_{k+i}^{(k)} := t mod B
```

```
r := t div B
finPour
```
$C_{k+m}^{(k)}$:= r

```
fin

procédure produit(a, b, c)
```
(* a = (a_{m-1}, $\cdots$, a_0), b = (b_{n-1}, $\cdots$, b_0) *)

```
debut
r := 0
```
Pour i := 0 jusqu'à m+n-1 faire C_i = 0

```
Pour k := 0 jusqu'à n-1 faire pro(k)
```
afficher (C_{m+n}, $\cdots$, C_0)

```
fin
```

Coût opérations : $n \times 4m$ opérations élémentaires.
ex : en base binaire si $a < 2^m$ on a $l(a) = \lfloor \log_2 a + 1 \rfloor$)
le produit ab est en $0(l^2)$, $l :=$ max(l(a), l(b))

5 Division de deux entiers :

$a = bq + r$ avec $0 < r < b$
1) $b < B$ ($l(b) = 1$)

```
procédure divrudi(a, b, q, r)
```
(* b $<$ B a = (a_{m-1}, $\cdots$, a_0) *)

```
debut
r := 0, i := m-1
```
Tant que i $\geq$ 0 faire
u := r $\times$ b + a_i

```
r := u mod b
```
q_i := u div b

```
i := i-1
finTantQue
```

afficher (q_{m-1}, $\cdots$, q_0) et r

```
fin
```

2) $q < B$, $b = b_{n-1}B^{n-1} + \cdots + b_0$, $a = a_n B^n + \cdots + a_0$
on pose $\widehat{q} = \min\left(B-1, \left\lfloor \frac{a_n \times B + a_{n-1}}{b_{n-1}} \right\rfloor\right)$
Proposition : Soit $q = a$ div b on a $q \leq \widehat{q}$
Démonstration :

$$b_{n-1}B^{n-1} \leq b < (b_{n-1}+1)B^{n-1}$$
$$a < a_n B^n + (a_{n-1}+1)B^{n-1}$$
$$q \leq \frac{a}{b} < \frac{a_n B^n + (a_{n-1}+1)B^{n-1}}{b_{n-1}B^{n-1}}$$
$$q b_{n-1} < a_n B + a_{n-1} + 1$$
$$q \leq \frac{a_n B + a_{n-1}}{b_{n-1}}$$

Proposition : Si $b_{n-1} \geq \left\lfloor \frac{B}{2} \right\rfloor$, on a $q \geq \widehat{q} - 2$

```
procédure DIVSIMP(A, B, Q, R)
(* 1 ≤ Q ≤ Base, b_{n-1} ≥ ⌊Base/2⌋ *)
debut
* recherche de q̂ *
q̂ := a_n * Base + a_{n-1} div b_{n-1}
Si q̂ ≥ Base + 1 alors q̂ := Base - 1
PRODUIT(B, q̂, E)
Tant que E > A faire
q̂ := q̂-1
PRODUIT(B, q̂, E)
finTantQue
R = E - A
imprimer q̂ et R
fin
```

3) procédure DIV(A, B, Q, R)

```
(* données A et B, sorties Q et R *)
debut
Si A < B alors Q := 0, R := A
Sinon
Si l(b) = 1 alors Divrudi(A, B, Q, R)
Sinon
début
facteur := 1
Si b_{n-1} < (Base div 2) alors
début
facteur := Base div (b_{n-1} + 1)
A := A * facteur
B := B * facteur
fin
L := l(A) - l(B)
C := (a_{n-1}, ···, a_L)
Pour I variant de L à 1 faire
DIVSIMP(C, B, Q_I, R)
C := R*Base + a_{I-1}
fin
```

6 Pgcd

6.1

fonction pgcd(a,b)

A := Max(a,b)
B := Min(a,b)
tant que B $\neq$ 0 faire
    C := A mod B

A := B
B := C

imprimer A

Première estimation du coût :

A l' etape 0 on a $a_0 = a\,;\, a_1 = b$

A l' etape i (variant de 1 a n) $a_{i-1} = a_i q_i + a_{i+1}$

A l' etape n $a_{n-1} = a_n q_n + \underbrace{a_{n+1}}_{=0}$

On a $q_i \geq 1 \quad i = 1, ..., n-1$ et $q_n \geq 2$ (sinon $q_n = 1$ et $a_{n-1} = a_n$)

Donc
$$\begin{aligned} a_0 &\geq a_1 + a_2 \\ &\vdots \\ a_{i-1} &\geq a_i + a_{i+1} \\ &\vdots \\ a_{n-1} &\geq \underbrace{2a_n \geq 2} \end{aligned}$$

$$a_n \geq \underbrace{1}_{F2},\ a_{n-1} \geq \underbrace{2}_{F3},\ a_{n-2} \geq \underbrace{3}_{F4},\ a_{n-3} \geq \underbrace{5}_{F5}\ \ldots,\ a_0 \geq F_{n+2}$$

Avec F_n la suite de Fibonacci ($F_n = F_{n-1} + F_{n-2}$)

$$F_{n+2} \sim \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+2}$$

puisque $a_0 \geq F_{n+2}$ on a $\log a_0 \geq (n+2) \log\left(\frac{1+\sqrt{5}}{2}\right) - \frac{1}{2}\log 5$

$$\log a_0 \geq n \log\left(\frac{1+\sqrt{5}}{2}\right) + 2 \log\left(\frac{1+\sqrt{5}}{2}\right) - \frac{1}{2}\log 5$$

donc $n \leq \frac{log\ a}{log(\frac{1+\sqrt{5}}{2})}$, soit $n \leq 1.218\ log_2 a$

A chaque étape il y a une division. Si a est 1 m-bits ($a < 2^m$), sachant qu'une division de deux nombres de m-bits (cas le plus défavorable), a un coût élémentaire d'au plus $O(m^2)$. Comme il y a m étapes, le coût est en $O(m^3) = O((log_2 a)^3)$.

Cette estimation du coût est moins précise puisqu'on ne tient pas compte que les a_i diminuent.

Deuxième estimation du coût

A l'etape i, le coût de la division
$$\begin{aligned} &= O((log_2 a_i + 1)(log_2(a_{i-1} - log_2 a_i + 1)) \\ &= O((log_2 a_i + 1)(log_2 q_i + 1) \\ &= O((log_2 a_i log_2 q_i) + \underbrace{O(log_2 q_i) + O(log_2 a_i) + 0(1)}_{\text{negligeable par rapport au } 1^{\text{er}} \text{ terme}} \end{aligned}$$

$$\text{coût} = \sum_{i=0}^{n}(O(log_2 a_i \, log_2 q_i)) = O(\sum_{i=0}^{n}(log_2 a_i \, log_2 q_i)) \quad \text{avec n = O(m)}$$

$a_i \leq a \qquad log_2\ a_i \leq log_2\ a \leq m$

$$\text{coût} = O(m \ \sum_{i=0}^{n} log_2 q_i) = O(m \ log_2 \prod_{i=0}^{n} q_i))$$

$$\begin{aligned}
a_0 &\geq a_1 q_1 \\
&\vdots \\
a_{i-1} &\geq a_i q_i \\
a_i &\geq a_{i+1} q_{i+1} \\
&\vdots \\
a_{n-1} &\geq a_n q_n \\
a_0 &\geq a_n \textstyle\prod_{i=1}^{n} q_i
\end{aligned}$$

D'où $\prod_{i=0}^{n} q_i \leq a$, le coût est donc en $O(m^2) = O((log_2 a)^2)$.

6.2

fonction pgcd(a,b) (* a > b *)

début
    si b = 0 alors a
            sinon pgcd (b, a mod b)
fin

6.3 algorithme binaire

$a = 2^k \alpha$ avec $(2, \alpha) = 1$
$b = 2^l \beta$ avec $(2, \beta) = 1$

$pgcd(a, b) = 2^{Min(k,l)} pgcd(\alpha, \beta)$

$pgcd(\alpha, \beta) \ = \ pgcd(\alpha - \beta, \beta) \qquad \alpha - \beta = 2^n \gamma$ avec $(2, \gamma) = 1$
$pgcd(\alpha, \beta) = pgcd(\gamma, \beta)$

fonction pgcd(a,b)
    k :=0;
    tant que a et b sont pairs faire
        a : = a/2 b := b/2
        k := k+1

    tant que a est pair faire a := a/2

    tant que b est pair faire b := b/2

    tant que $a \neq b$ faire

$$\left\{\begin{array}{rcl} a & := & max(a,b) \\ b & := & min(a,b) \end{array}\right\} \text{instruction non sequentielle}$$

a := a -b

tant que a pair faire
a := a div 2

imprimer $2^k \times a$
<u>coût</u> : $O(log_2 a)$

7 Algorithme d'Euclide généralisé

Cet algorithme utilise l'identité de Bezout :

$$\forall a, b\ \exists u, v / \quad au + bv = \delta \text{ où } \delta = pgcd(a,b)$$

départ : $\left\{\begin{array}{rcl} a & = & a \times 1 + b \times 0 \\ b & = & a \times 0 + b \times 1 \end{array}\right.$

arrivée : $\left\{\begin{array}{rcl} \delta & = & a \times u + b \times v \\ 0 & = & a \times u_1 + b \times v_1 \end{array}\right.$

intermédiaire : $\left\{\begin{array}{rcl} a_i & = & a \times \alpha_i + b \times \beta_i \\ a_{i+1} & = & a \times \alpha_{i+1} + b \times \beta_{i+1} \end{array}\right.$

<u>initial</u> (1) : $\left\{\begin{array}{rcl} A_0 & = & A \times S_0 + B \times T_0 \\ B_0 & = & A \times S_1 + B \times T_1 \end{array}\right.$ relation à une étape de l'algorithme $B_0 = A_1$

$$a_i = a_{i+1} q_{i+1} + a_{i+2}$$

étape suivante

$$\begin{array}{rcl} a_{i+1} & = & a \times \alpha_{i+1} + b \times \beta_{i+1} \\ a_{i+2} & = & a_i - a_{i+1} q_{i+1} = a(\alpha_i - q_{i+1}\alpha_{i+1}) + b(\beta_i - q_{i+1}\beta_{i+1}) \end{array}$$

donc si à une étape on a les relations (1), à l'étape suivante on aura :
avec Q le quotient de la division euclidienne de A_0 par A_1

$$\begin{array}{rcl} Q & := & A_0 \text{ div } A_1 \\ (A_0, B_0) & := & (B_0, A_0 - B_0 \times Q) \\ (S_0, T_0) & := & (S_1, T_1) \\ (S_1, T_1) & := & (S_0 - S_1 \times Q, T_0 - T_1 \times Q) \end{array}$$

Algorithme

$A_0 := Max(a,b)$
$A_1 := Min(a,b)$
$S_0 := 1 \quad S_1 := 0$
$T_0 := 0 \quad T_1 := 1$
tant que $A_1 \neq 0$ faire
    $Q := A_0$ div A_1
    $(A_0, A_1) := (A_1, A_0 - A_1 \times Q)$
    $(S_0, S_1) := (S_1, S_0 - S_1 \times Q)$
    $(T_0, T_1) := (T_1, T_0 - T_1 \times Q)$

imprimer (S_0, T_0, A_0)
    S_0 represente u
    T_0 represente v
    A_0 represente δ

Applications

1. Résolution de $ax + by = c \quad a, b, c \in \mathbb{Z}$ avec $x, y \in \mathbb{Z}$ inconnues :
il existe des solutions si et seulement si $pgcd(a,b)$ divise c.
2. Résolution de $ax \equiv 1 \bmod m$, inverse de a dans $\mathbb{Z}/m\mathbb{Z}$.

8 Fonction φ d'Euler

On considère l'anneau $\mathbb{Z}/m\mathbb{Z}$ et l'ensemble de ses éléments inversibles ou unités, noté $(\mathbb{Z}/m\mathbb{Z})^* = E_m$ et on pose $\varphi(m)$=Card E_m (le nombre d'éléments de E_m).
Si p est premier on a $\varphi(p) = p - 1$ et $\varphi(p^\alpha) = p^{\alpha-1}(p-1)$.
Propriété : φ est multiplicative $\varphi(ab) = \varphi(a)\varphi(b)$ si $pgcd(a,b) = 1$.
Corollaire : $\varphi(m) = \varphi(p_1^{\alpha_1} \cdots p_k^{\alpha_k}) = p_1^{\alpha_1 - 1}(p_1 - 1) \cdots p_k^{\alpha_k - 1}(p_k - 1)$.
Identité d'Euler : $a^{\varphi(m)} \equiv 1 \bmod m$ si $pgcd(a,m) = 1$.
Corollaire :
1. Identité de Fermat : Si p premier et a non multiple de p alors $a^{p-1} \equiv 1 \bmod p$.
2. Calcul de l'inverse de a lorsque $pgcd(a,m) = 1$: $a^{-1} = a^{\varphi(m)-1}$.

9 Théorème du reste Chinois dans $\mathbb{Z}$

Théorème : Soient $m_1, m_2, \cdots, m_n$ n entiers premiers 2 à 2 alors le système $u \equiv u_j \bmod m_j$ $j = 1, \cdots, n$ a une et une seule solution en u dans tout intervalle $[a, a + \prod_{i=1}^n m_i[$.
Exemple :

$$\begin{cases} u \equiv 2 \bmod 3 \\ u \equiv 3 \bmod 5 \end{cases}$$

Dans l'intervalle $[0, 15[$ la solution est $u = 8$.
Preuve :
Soit $M = \prod_{i=1}^n m_i$ posons $M_i = M/m_i$ on a $pgcd(M_i, m_i) = 1$
donc il existe N_i tel que $N_i M_i \equiv 1 \bmod m_i$ (N_i inverse de M_i mod m_i)
posons $u = u_1 N_1 M_1 + \cdots + u_n N_n M_n$

alors $u \bmod m_j \equiv u_j \underbrace{N_j M_j}_{=1} \bmod m_j \equiv u_j \bmod m_j$.

Algorithme

$M := 1$
$u := u_1 \bmod m_1$
pour $k = 2$ jusqu'à n faire
  $M := M \times m_k$, C :=Inverse de M mod m_k
  $(S := ((u_k - u \bmod m_k) \times C) \bmod m_k$
  $u := u + S \times M$

retourner u

Remarques :
1. Il existe une solution u telle que $0 \leq u < M$.
2. On peut rajouter une équation à la fin des calculs sans tout refaire dans l'algorithme.
3. $u = S_1 + S_2 m_1 + S_3 m_1 m_2 + \cdots + S_n m_1 m_2 \cdots m_{n-1}$

Application de l'exemple :
$M = 1$
$U = 2$
$M = 3$
$C = Inverse((3, 5) = 2$
$S = (3 - 2) \times 2 = 2$
$U = 2 + 2 \times 3 = 8$