Chapitre 1

La division euclidienne et ses conséquences

1.1 La division euclidienne

Division euclidienne pour les entiers positifs

Théorème 1.1.1. *Pour $a \in \mathbb{N}$, $b \in \mathbb{N}^*$, il existe un unique couple d'entiers (q, r) avec $a = bq + r$ et $0 \leq r < b$.*

Méthodes algorithmiques :

Méthode naïve :

Entrer $a \geq 0$ et $b > 0$;
$n \leftarrow 0$; $r \leftarrow a$;
Tant que $(r > b)$ **faire**
    $r \leftarrow r - b$;
    $q \leftarrow q + 1$;
Fait
Retourner q et r ;

Algorithme 1: Division euclidienne, méthode naïve

Le nombre de boucle de cet algorithme est q ; à chaque étape il y a une soustraction et une comparaison ; pour b fixé la complexité croît linéairement avec a.

Remarque 1.1.2. Si on mesure la taille de l'entier a par le nombre de chiffres de son développement en binaire : $t = \log_2(a)$, alors la *complexité* croît exponentiellement avec la taille de a.

Recherche dichotomique

Entrer $a \geq 0$ et $b > 0$;
$n \leftarrow 0$;
Tant que $(2^n b \leq a)$ **faire**
  $n \leftarrow n + 1$;
Fait
$\alpha \leftarrow 2^{n-1}$;$\beta \leftarrow 2^n$;
Pour k **de** 1 **à** n-1 **faire**
  $\gamma = \frac{\alpha+\beta}{2}$;
  Si $(\gamma b \leq a)$ **Alors**
    $\alpha \leftarrow \gamma$;
  Sinon
    $\beta \leftarrow \gamma$;
  Fin Si
Fin Pour
Retourner $q = \alpha$ et $r = a - bq$;

Algorithme 2: Division euclidienne, recherche dichotomique

Dans le premier calcul on considère la suite géométrique 2^n. En sortie de la boucle *Tant que*, on a un encadrement du quotient q :

$$\alpha = 2^{n-1} \leq q < 2^n = \beta \ .$$

Pour le deuxième calcul, on fait une recherche dichotomique ; lors de la boucle indexée par k, la taille de l'encadrement est : 2^{n-1-k}. En sortie : $k = n-1$; on a un encadrement de taille 1 qui est strict à droite, et donc $q = \alpha$.

Exercice 1.1.3. Ecrire un algorithme qui évite la multiplication.

Cet algorithme a une complexité qui est logarithmique en a : $n \sim \frac{\log_2(a)}{\log_2(b)} = O(\ln(a))$. La complexité, comme fonction de la taille t de a est donc linéaire.

Division euclidienne pour les entiers relatifs

Théorème 1.1.4. *Pour $a \in \mathbb{Z}$, $b \in \mathbb{Z}^*$, il existe un unique couple d'entiers (q, r) avec $a = bq + r$ et $0 \leq r < |b|$.*

Exercice 1.1.5. Modifier si besoin les algorithmes précédents pour effectuer la division euclidienne des entiers relatifs.

1.2 Sous-groupes de $\mathbb{Z}$

Définition 1.2.1. On appelle sous-groupe de $\mathbb{Z}$ tout sous-ensemble H qui contient 0 et qui est stable par addition et soustraction.

Exemples : $n\mathbb{Z}$, $a\mathbb{Z} + b\mathbb{Z}$, l'intersection de deux sous-groupes.

Théorème 1.2.2. *Tout sous-groupe de $\mathbb{Z}$ est de la forme $n\mathbb{Z}$, avec $n \in \mathbb{N}$.*

On dit que n est générateur du sous-groupe.

1.3 Diviseurs

Proposition 1.3.1. *Pour $a \in \mathbb{Z}$ et $b \in \mathbb{Z}$, il y a équivalence entre a divise b et*

$$b\mathbb{Z} \subset a\mathbb{Z} \ .$$

Proposition 1.3.2. *a) Le générateur positif d de $a\mathbb{Z} + b\mathbb{Z}$ est le PGCD de a et b : c'est le plus grand diviseur commun à a et b.*
b) Les diviseurs communs à a et b sont les diviseurs du PGCD.

Proposition 1.3.3. *a) Le générateur positif de $a\mathbb{Z} \cap b\mathbb{Z}$ est le PPCM de a et b : c'est le plus petit multiple commun à a et b.*
b) Les multiples communs à a et b sont les multiples du PPCM.

1.4 L'algorithme d'Euclide

Proposition 1.4.1. *Soient : $a \in \mathbb{Z}$, $b \in \mathbb{Z}$. On a pour tout $m \in \mathbb{Z}$:*

$$PGCD(a, b) = PGCD(b, a - mb) \ .$$

On peut en particulier appliquer la proposition précédente au cas où $m = q$ est le quotient dans la division euclidienne de a par b.

```
Fonction PGCD(a, b) ;
a ← |a| ; b ← |b| ;
Si (b > a) Alors
   | a ↔ b ;
Fin Si
r ← a mod b (reste de la division euclidienne) ;
Si (r est nul) Alors
   | Retourner b ;
Sinon
   | Retourner PGCD(b, r) ;
Fin Si
```

Algorithme 3: Algorithme d'Euclide, forme récursive

```
Fonction PGCD(a, b) ;
a ← |a| ; b ← |b| ;
Si (b > a) Alors
   | a ↔ b ;
Fin Si
Tant que (b est non nul) faire
   | r ← a mod b (reste de la division euclidienne) ;
   | a ← b ;b ← r ;
Fait
Retourner a ;
```

Algorithme 4: Algorithme d'Euclide, forme non récursive

1.5 Théorème de Bezout

Rappel : Deux entiers relmatifs sont premiers entre eux si et seulement si leur PGCD est égal à 1. Un entier naturel p est premier si et seulement s'il a exactement deux diviseurs positifs : 1 et p.

Théorème 1.5.1. *Deux entiers a et b sont premiers entre eux si et seulement s'il existe un couple (u, v) tel que $au + bv = 1$.*

Corollaire 1.5.2 (théorème de Gauss)**.** *Soient a, b et c trois entiers relatifs. Si a divise le produit bc et est premier avec b, alors a divise c.*

Corollaire 1.5.3 (théorème d'Euclide)**.** *Si un entier premier divise un produit, alors il divise un des facteurs.*

Equation $ax + by = c$**.**

Proposition 1.5.4. *a) L'équation* $ax + by = c$ *a des solutions si et seulement si le PGCD de* a *et* b *divise* c*.*
b) Lorsque a *et* b *sont premiers entre eux, si* (x_0, y_0) *est une solution, alors les solutions sont :* $(x = x_0 - kb, y = y_0 + ka),\ k \in \mathbb{Z}$*.*

1.6 Algorithme d'Euclide étendu aux coefficients de Bezout

Fonction PGCDE(a, b) ; *[la fonction retourne 3 entiers]*
$a \leftarrow |a|$; $b \leftarrow |b|$;
Si $(b > a)$ **Alors**
  $a \leftrightarrow b$;
Fin Si
Effectuer la division euclidienne de a par b ; $r \leftarrow$ reste ;$q \leftarrow$ quotient ;
Si (r est nul) **Alors**
  Retourner $(b, 0, 1)$;
Sinon
  $(d, u', v') \leftarrow PGCDE(b, r)$;
  $u \leftarrow v'$; $v \leftarrow (u' - qv')$;
  Retourner (d, u, v) ;
Fin Si

Algorithme 5: Algorithme d'Euclide, forme récursive

Exercice 1.6.1. Ecrire un algorithme d'Euclide étendu non récursif.

Etude de complexité : Soit $t = \ln(\max(|a|; |b|))$. On montre (théorème de Lamé) que le nombre de boucles récursives est un $O(t)$, et que la complexité est un $O(t^2)$. Il existe des algorithmes améliorés, de complexité *quasi-linéaire*.

1.7 Décomposition en facteurs premiers

Théorème 1.7.1. *Tout entier naturel supérieur ou égal à* 2 *s'écrit de manière unique comme un produit de facteurs premiers.*

Exercice 1.7.2. Ecrire un algorithme pour tester si un nombre est premier.

Exercice 1.7.3. Ecrire un algorithme pour décomposer en facteurs premiers. On pourra utiliser un tableau T contenant les nombres premiers successifs jusqu'à $T[N]$. On traitera en entrée les entiers inférieurs ou égaux à $T[N]^2$; en sortie on obtient les facteurs premiers avec leur exposant.