Mathématiques appliquées à l'informatique
2 – Arithmétique : division, divisibilité et congruence

Bertrand LIAUDET

SOMMAIRE

Dernière édition : janvier 2017

CHAPITRE 2 – DIVISION ET DIVISIBILITE

0 - Préambule

Principes

Partir d'une application pour remonter à l'objet mathématique théorique qu'on étudie abstraitement ensuite.

Ensuite on fait des exercices théoriques (des gammes), des exercices appliqués et des traductions algorithmiques.

Le but est de rentrer dans une logique de mathématiques appliquées, c'est-à-dire : les maths comme un outil et pas une abstraction creuse.

Objectif : ne pas avoir peur des maths ! Les voir comme un outils pratique qui permet de régler des problèmes efficacement.

Références

Mathématique pour l'informatique – BTS SIO – Dunod – 2015 : Chapitre 1, pp. 3-33.

Méthodes mathématiques pour l'informatique – IUT-Licence-Ecole d'ingénieurs-CNAM – Dunond 2013.

Le problème : Algorithme de César – Codage Affine

Cryptage, chiffrement, codage

Le cryptage ou chiffrement est un procédé pour rendre la compréhension d'un document impossible sans avoir la clé de déchiffrement. On parle aussi de chiffrement ou de codage (mais le codage est une notion plus large).

Algorithme de César

L'algorithme de César est un algorithme de cryptage. C'est un algorithme simple.

Les lettres du texte sont numérotées de 0 à 25 (a=0, b=1, …, z=25). Le chiffrement consiste à remplacer chaque lettre du texte par une autre en appliquant un décalage « d », d étant un entier compris entre 0 et 25. Par exemple, si d vaut 5, a est remplacé par f, b par g, etc. z est remplacé par e.

Pour décoder un texte, il faut connaître le décalage. Toutefois, si on en connaît pas la valeur de d, on peut appliquer le principe suivant : dans un texte, la lettre « e » est le plus souvent la plus représentée. Ainsi, on peut trouver le décalage à partir du texte codé.

1. Chercher « manuellement » à décoder le texte suivant : « tew wm jegmpi uyi gipe pi gsheki hi giwev »
2. Ecrire un programme qui permet de coder et décoder des textes. Pour cela, on écrira :

- Une fonction coderMessage (message, d) qui renvoie le message codé. Comment utiliser cette fonction pour décoder ?
- Une fonction frequenceDesLettres (message) qui renvoie un tableau avec la fréquence de chaque lettre de l'alphabet dans le message.
- Une fonction calculDecalage (message) qui renvoie le décalage probable du message
- Le programme qui permet de lire un texte codé puis de l'afficher décodé.

Codage affine

Le codage affine (ou chiffrement affine) est une méthode assez proche de l'algorithme de César mais plus subtile puisque le déchiffrement est plus complexe à réaliser.

On part de lettres numérotées de 0 à 25. Le codage affine correspond à une fonction :

f(numéro_lettre_codée) = (a * numéro_lettre + b) % 26 (% c'est le modulo, c'est-à-dire le reste de la division entière). On fait un modulo 26 pour récupérer un entier compris entre 0 et 25 qui correspond aux lettres de « a » à « z ».

1. Codage affine avec a=3 et b=11
   - Comment sont codés G et S ?
   - Remplir un tableau de 4 lignes et 26 colonnes. Ligne 1 : Les 26 lettres de l'alphabet. Ligne 2 : le numéro des lettres (de 0 à 25). Ligne 3 : le numéro codé des lettes en appliquant la fonction affine. Ligne 4 : lettre codée correspondant au numéro codé.
   - Quel est le mot codé par VBUTSB
   - Déterminer la fonction de décodage, c'est-à-dire calculer numéro_lettre en fonction de numéro_lettre_codée. Cela passe par l'inverse du modulo 26 (voir le cours pour l'inverse du modulo). Décoder la lettre « L ».
2. Décodage, sachant que E est codé par I et que V est codé par T :
   - Ecrire les deux congruences vérifiées par a et b (voir le cours).
   - Quelle est la fonction de décodage ? On utilisera le programme excel de calcul de l'inverse du modulo.
3. Comparer le codage affine et le codage de César.
4. Ecrire un programme qui permet de coder et décoder des textes. Pour cela, on écrira :
   - Une fonction codageAffineMessage (message, a, b) qui renvoie le message codé. Comment utiliser cette fonction pour décoder ?
   - Une fonction inverseCodageAffine(a, b, a_inv, b_inv) qui ne renvoie rien mais dont les deux paramètres a_inv et b_inv sont en sortie et correspondent aux paramètres de la fonction de décodage
   - Le programme qui permet de lire un texte codé puis de l'afficher décodé.

Les exercices

1 - Division

2 - Divisibilité et nombres premiers

1. Pour chacun des nombres proposés, dire si le nombre est premier. S'il l'est quel est le dernier diviseur testé ? S'il ne l'est pas, quel est son plut petit diviseur ?

   107 - 161 – 179 – 241 – 311 – 323 – 437 – 563 – 677 – 779

2. Ecrire l'algorithme d'une fonction qui renvoie le plus petit diviseur d'un nombre entier et 0 si le nombre est premier.

3. Ecrire la liste des diviseurs des nombres entiers suivants en utilisant leurs décompositions en facteurs premiers : 48, 135, 1617

4. Ecrire l'algorithme qui affiche les diviseurs à partir de la décomposition en nombres premiers (la série des nombres premiers et la puissance qui va avec).

5. Quel est le PGCD de 315 et 1171. Appliquez la méthode d'Euclide puis la méthode de la soustraction. Idem avec 882 et 540.

6. Ecrivez l'algorithme d'Euclide puis celui de la soustraction sous excel.

7. Ecrivez l'algorithme d'Euclide puis celui de la soustraction en pseudo-code. Eventuellement codez-le en c ou en python.

3 – Congruence

8. Ecrire un programme excel permettant de calculer l'inverse du modulo avec un algorithme naïf

   Il suffit, par exemple, de se doter d'une colonne qui calcule ax, d'une colonne qui calcule ax%n et d'une colonne qui vérifie si le résultat précédent est égal à 1.

9. Ecrire un algorithme qui calcule l'inverse du modulo en vous inspirant de votre programme Excel.

10. Ecrire un programme excel qui applique l'algorithme d'Euclide étendu qu'on trouve : ici.

    Essayer de comprendre le principe de résolution de l'algorithme (ce n'est pas simple !)

1 - Division

Division euclidienne

Les entiers naturels sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, etc.

Soit A, B, Q et R $\in$ N

Effectuer la division euclidienne de A(appelé dividende) par B (appelé diviseur) non nul c'est déterminer les uniques entiers Q (appelé quotient) et R (appelé reste) tels que :

$$A = B*Q + R$$
$$\text{avec } R < B$$

On peut obtenir les résultats en posant la division ou avec une calculatrice.

Division entière : div

Présentation

L'opération de division entière s'écrit : a div b.

Son résultat est le quotient de la division euclidienne.

Si on travaille avec des entiers, on peut aussi écrire a / b

Par exemple : 9 div 4 = 2, 11/3 = 3

En python, on écrit a // b

Propriété

Soit A, B non nul, Q, R et n $\in$ N

$$A / B = Q$$
$$\Leftrightarrow$$
$$(A + n*B) / B = Q + n$$

Par exemple :

11 / 5 = 2

(11 +3*5) / 5 = 2+3

Modulo : mod ou %

Présentation

L'opération de modulo s'écrit : a mod b ou a%b.

Son résultat est le reste de la division euclidienne.

Par exemple : 9 mod 4 = 1, 11%3 = 2

Propriété

Soit A, B non nul, Q, R et n ∈ N

$$A \% B = R$$
$$\Leftrightarrow$$
$$(A + n*B) \% B = R$$

Par exemple :

11 % 5 = 1

(11 +3*5) % 5 = 1

Propriété 2

Soit A, B non nul, Q, R et X ∈ N

$$A \% B = R$$
$$\Leftrightarrow$$
$$(A + X) \% B = (R + X) \% B$$

Par exemple :

11 % 5 = 1

(11 + 7) % 5 = (1 +7) % 5

2 - Divisibilité et nombres premiers

Nombre premier

Définition

Un nombre est premier s'il n'a que 2 diviseurs : 1 et lui-même.

0 n'est pas premier car il a une infinité de diviseurs. 1 n'est pas premier car il n'a qu'un seul diviseur.

Série des premiers nombres premiers

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, etc.

Propriété

Le plus grand diviseur d'un nombre entier n est $\leq \sqrt{n}$

Algorithme

De ce fait, pour savoir si un nombre n est premier, on vérifie qu'il n'est divisible par aucun nombre $\leq \sqrt{n}$

Décomposition en facteurs premiers

Définition

Un nombre qui n'est pas premier peut s'écrire comme un produit de nombres premiers.

Tout nombre peut se décomposer en facteurs premiers :
$n = a^i * b^j * c^k * \ldots$

Exemple = $54 = 6 * 9 = 2 * 3 * 3 * 3 = 2^1 * 3^3$

Algorithme

Pour décomposer un nombre en facteurs premiers, on divise le nombre par son plus petit diviseur premier et on recommence jusqu'à ce que le diviseur soit égale au dividende.

Tous les diviseurs d'un nombre

Tout nombre peut se décomposer en facteurs premiers :

$n = a^i * b^j * c^k * \ldots$

Les diviseurs de n sont de la forme : $a^{ii} * b^{jj} * c^{kk} * \ldots$ avec $0 \leq ii \leq i, 0 \leq jj \leq j, 0 \leq kk \leq k$, etc.

Exemple = $54 = 2^1 * 3^3$

Les diviseurs de 54 sont sont de la forme : $2^{ii} * 3^{jj}$ avec $0 \leq ii \leq 2, 0 \leq jj \leq 2$

Les diviseurs de 54 sont : $2^0 * 3^0 = \mathbf{1}$, $2^0 * 3^1 = \mathbf{3}$, $2^0 * 3^2 = \mathbf{9}$, $2^0 * 3^3 = \mathbf{27}$, $2^1 * 3^0 = \mathbf{2}$, $2^1 * 3^1 = \mathbf{6}$, $2^1 * 3^2 = \mathbf{18}$, $2^1 * 3^3 = \mathbf{54}$.

Soit : 1, 3, 9, 27, 2, 6, 18, 54

PGCD

Définition

Soit 2 nombres entiers a et b.

Les 2 ensembles de leurs diviseurs ont au moins un nombre en commun : 1.

Le plus grand des nombres en commun de ces deux ensembles est appelé : Plus Grand Commun Diviseur.

On le note PGCD(a,b)

Nombres premiers entre eux

Si PGCD (a, b) = 1

Alors

a est b sont premier entre eux.

PGCD : produit de facteurs premiers

PGCD (a, b) =

produits des facteurs commun de la décomposition en facteurs premier de a et b,

chaque facteur étant affecté du plus petit exposant avec lequel il figure dans les deux décomposition.

Propriété – Algorithme d'Euclide

Si a > b

Alors

PGCD (a, b) = PGCD (a%b, b)

Exemple : PGCD(98, 42) = PGCD(42, 98%42) = PGCD(42,14)

PGCD(42,14) = PGCD(14, 0) = 14 (0 est divisible par tout n >0).

Cette propriété permet de calculer le PGCD de façon simple. Cela correspond à l'algorithme d'Euclide.

Autre propriété – Autre algorithme

Si a > b

Alors

PGCD (a, b) = PGCD (a-b, b)

Exemple : PGCD(98, 42) = PGCD(98-42, 42) = PGCD(56,42)

Cette propriété permet aussi de calculer le PGCD de façon simple.

3 - Congruence

Définition

Formulation

A est congru à B modulo N

⇔

A%N = B%N

⇔

A ≡ B[N]

Exemple : 7 ≡ 4[3] ⇔7%3 = 4%3

Signification

A et B ont le même reste si on les divise par N

Commutativité

A ≡ B[N]

⇔

B ≡ A[N]

⇔

A et B sont congrus à N

Exemple : 7 ≡ 4[3] ⇔ 4 ≡ 7[3]

- ***Exercice : démonstration de la propriété***

  En passant par les modulos

Propriétés

L'observation des propriétés permet de se familiariser avec la notion de congruence.

Propriété 1 : élément neutre et congruence

Si on ajoute N à un des 2 nombres congrus à N, ils le restent.

$$A \equiv B[N]$$
$$\Leftrightarrow$$
$$A+N \equiv B[N]$$

Exemple : $7 \equiv 4[3] \Leftrightarrow 7+3 \equiv 4[3] \Leftrightarrow 10 \equiv 4[3]$

- ***Exercice : démonstration de la propriété***

  En passant par la définition : A%N = B%N et par une propriété des modulos : A%N = (A+N)%N

Propriété 2 : addition et congruence

Si on ajoute une même valeur à 2 nombres congrus, ils le restent.

$$A \equiv B[N]$$
$$\Leftrightarrow$$
$$\forall p \in N, A+p \equiv B+p[N]$$

Exemple : $7 \equiv 4[3] \Leftrightarrow 7+12 \equiv 4+12[3] : 19 \equiv 16[3]$

- ***Exercice : démonstration de la propriété***

  En passant par la définition : A%N = B%N.

Propriété 3 : multiplication et congruence

Si on multiplie 2 nombres congrus par une même valeur, ils le restent.

$$\text{Si } A \equiv B[N]$$
$$\text{Alors}$$
$$\forall p \in N, pA \equiv pB[N]$$

Exemple : si $7 \equiv 4[3]$ alors $7*5 \equiv 4*5[3] : 35 \equiv 20[3]$

- ***Exercice : démonstration de la propriété***

  En passant par la définition : A%N = B%N.

Propriété 4 : puissance et congruence

Si on passe 2 nombres congrus à la même puissance, ils le restent.

$$\text{Si } A \equiv B[N]$$
$$\text{Alors}$$
$$\forall p \in N, A^p \equiv B^p [N]$$

Exemple : si $7 \equiv 4[3]$ alors $7*7 \equiv 4*4[3] : 49 \equiv 16[3]$

Propriété 5 : 2 paires et congruence

Si les valeurs de 2 paires sont congrues à N, la paire obtenue par addition, soustraction ou multiplication entre elles des deux paires est congru à N.

Si $A \equiv B[N]$ et $C \equiv D[N]$
Alors
$A+C \equiv B+D[N]$
et $A-C \equiv B-D[N]$
et $AC \equiv BD[N]$

Exemples :

$7 \equiv 16[3]$ et $23 \equiv 5[3]$

donc :

$7+23 \equiv 16+5\ [3] \Leftrightarrow 30 \equiv 21\ [3]$

$7-23 \equiv 16-5[3] \Leftrightarrow -16 \equiv 11\ [3] \Leftrightarrow 16-16 \equiv 16+11\ [3] \Leftrightarrow 0 \equiv 27\ [3]$

$7+23 \equiv 16*5[3] \Leftrightarrow 161 \equiv 80\ [3] \Leftrightarrow 53*3+2 \equiv 26*3+2\ [3]$

- ***Exercice : démonstration de la propriété***

  En passant par la définition : A%N = B%N et C%N = D%N

Propriété 6 : congruence et division euclicienne

La soustraction de 2 nombres congrus à N est divisible par N

$A \equiv B[N]$
$\Leftrightarrow$
$(A-B)\ \%\ N = 0$
$\Leftrightarrow$
$A - B \equiv 0[N]$

Exemple : $16 \equiv 4[3] \Leftrightarrow (16-4)\ \%\ 3 = 0$

- ***Exercice : démonstration de la propriété***

  En passant par la propriété 2 : addition et congruence

  En passant par la définition : A%N = B%N.

Propriété 7 : congruence et modulo

$A\ \%\ N = R$
$\Leftrightarrow$
$A \equiv R[N]$

Exemple : si $25 = 3 * 8 + 1$ alors $25 \equiv 1[8]$ et $25 \equiv 1[3]$

- ***Exercice : démonstration de la propriété***

  En passant par la définition : A%N = R%N.

  Et en passant par le fait qu'on ait une division euclidienne.

- ***Cas particulier***

Si A%N=1 alors A ≡ 1[N]

Ce cas est intéressant pour l'inverse du modulo.

Exercices

Faites les divisions euclidiennes de 200 et de 900 par 13.

Traduisez-les en congruence (propriété 7).

En utilisant les propriétés des congruences, cherchez X dans les formules suivantes :

$200 + 900 \equiv X\ [13]$ (propriété 2).

$200 \times 900 \equiv X\ [13]$ (propriété 3).

$200^2 \equiv X\ [13]$ (propriété 4).

$900^3 \equiv X\ [13]$ (propriété 4).

$200^4 + 900^6\ \%\ 13 = X$ (propriétés 4 et 5).

Inverse du modulo

Définition

Soit a % n,
L'inverse de a % n c'est
le nombre entier $\mathbf{a^{-1} < n}$ **tel que**
$\mathbf{(a * a^{-1}) \% n = 1}$
$\Leftrightarrow (a * a^{-1}) \equiv 1[n]$ (propriété 7)

Remarque : on écrit a^{-1} car dans R, $a^{-1} = 1/a$ et donc $a * a^{-1} = 1$. On a donc bien $(a * a^{-1}) \equiv 1[n]$

On écrit : $\mathbf{a^{-1} \% n}$
pour dire : **l'inverse de a modulo n**

Exemples

- ***Soit 4 % 9 = 4***

  On cherche 4^{-1} % 9, l'inverse de 4 modulo 9. On l'appelle X.
  On cherche X tel que 4X % 9 =1
  $\Leftrightarrow 4X \equiv 1[9]$ (propriété 7)
  $\Leftrightarrow 4X - 1 \equiv 0[9]$ (propriété 2)
  On cherche les multiples de 4 qui, quand on leur ôte 1, sont divisibles par 9.
  Réponse : X=7 : en effet, 4*7=28, 28-1=27, 27%9 =0.

- ***Soit 8 % 27 = 8***

  on cherche 8^{-1} % 27, l'inverse de 8 modulo 27. On l'appelle X.
  On cherche X tel que 8X % 27 =1
  $\Leftrightarrow 8X \equiv 1[27]$ (propriété 7)
  $\Leftrightarrow 8X - 1 \equiv 0[27]$ (propriété 2)
  Donc on cherche X tel que (8*X – 1) %27 = 0
  8-1=7, 16-1=15, 24-1=23, etc, 17*8=136-1=135 % 27=0.
  Réponse : X=17 : il n'y a pas de méthode de calcul simple pour le trouver !

Propriété

Si $(a^{-1}$ % n) existe
$\Leftrightarrow$
a et n sont premiers entre eux
$\Leftrightarrow$
PGCD(a, n) = 1.

Comment trouver l'inverse du modulo ?

- ***Première approche : l'algorithme naïf***

  On cherche A^{-1} % n : l'inverse de A modulo N.

  Donc on cherche X tel que (AX – 1) %N = 0

  Donc chercher A^{-1}, c'est chercher le plus petit multiple de A dont la valeur précédente (-1) est divisible par N.

  Un tableur excel permet de trouver le résultat facilement.

  On peut aussi coder une fonction en python par exemple.

- ***Deuxième approche : l'algorithme d'Euclide étendu***

  L'algorithme d'Euclide étendu (voir ici) permet de trouver le résultat plus rapidement mais il n'est pas facile à interpréter !

- ***Exercice***

  Coder en python l'algorithme naïf et l'algorithme d'Euclide étendu. Pour l'algorithme naïf, vous devez trouver le code. Pour l'algorithme d'Euclide, vous pouvez utiliser l'algorithme proposé ci-dessus.

  Mettez un compteur dans chaque boucle et comparez le nombre de tour dans un algorithme et dans l'autre.

Fonction affine, codage et décodage

Fonction affine

Une fonction affine est une fonction de R dans R de la forme

$$f(x)=y=a*x+b,$$

avec x, y, a, b appartenant à R.

Fonction affine et codage

Soit une fonction affine f de E dans E, E étant un intervalle de N

$$f(x)=y= (a*x+b) \% \text{ card}(E)$$

avec x, y, a, b appartenant à E.

On peut définir la fonction avec une congruence :

$$y \equiv a*x+b[\text{card}(E)]$$

Une telle fonction peut être appelée « fonction de codage ».

Décodage

Le décodage va consister à **chercher x en fonction de y.**

On part de la fonction de codage :	$y \equiv a*x+b\ [\text{card}(E)]$
Propriété 2 : on ajoute –b	$\Leftrightarrow y - b \equiv a*x\ [\text{card}(E)]$
Prop. 3 et inv. du modulo : on multiplie par a^{-1}	$\Leftrightarrow a^{-1}*(y - b) \equiv x\ [\text{card}(E)]$
Commutativité	$\Leftrightarrow x \equiv a^{-1}*(y – b)\ [\text{card}(E)]$

Exemple : codage de l'alphabet

Codage de l'alphabet : on a 26 lettres qu'on numérote de 0 à 25. On a donc E = [0 ; 26 [

On se dote de fonction de codage : f(x) = y = a*x + b tel que $y \equiv a*x+b[26]$ avec $x, y, a, b \in E$

- ***Exemple : a=5 et b=3 – fonction de codage***

f(x)=y=(5x+3) %26

$y \equiv 5x+3[26]$

Si x=10 (lettre « k »), y = 53%26 = 1 (lettre « b »)

- ***Fonction de décodage***

On part de la fonction de codage	$y \equiv 5x+3\ [26]$
On arrive à :	$\Leftrightarrow x \equiv 5^{-1}*(y – 3)\ [26]$

5^{-1} % 26 = 21 (on trouve le résultat en utilisant un algorithme)

donc	$\Leftrightarrow x \equiv 21 * (y -3)\ [26]$
	$\Leftrightarrow x \equiv 21y - 63\ [26]$
On applique la propriété 1 : élément neutre	$\Leftrightarrow x \equiv 21y -63 + 3*26\ [26]$
	$\Leftrightarrow x \equiv 21y +15\ [26]$