Mathématiques appliquées à l'informatique
Arithmétique :
division, divisibilité, congruence
chiffrement

Bertrand LIAUDET

SOMMAIRE

SOMMAIRE 1

DIVISION ET DIVISIBILITE 2

Références 2

Exercice d'introduction 2

Le problème : Algorithme de César – Codage Affine 2

Chiffrement mono alphabétique 4

Chiffrements plus subtils 4

1 - Division 6

Division euclidienne = division entière 6

Division entière : opérateur « div » 6

Modulo : opérateur « mod » ou % 7

Exercices 7

2 - Divisibilité et nombres premiers 8

Nombre premier 8

Décomposition en facteurs premiers 9

Tous les diviseurs d'un nombre 10

PGCD 11

3 - Congruence 13

Définition 13

Propriétés du modulo 14

Propriétés de la congruence 15

Inverse du modulo 18

Fonction affine, codage et décodage 20

Dernière édition : janvier 2018

DIVISION ET DIVISIBILITE

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.

http://exo7.emath.fr

http://exo7.emath.fr/cours/ch_crypto.pdf

Exercice d'introduction

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 codage à la place de cryptage ou chiffrement, mais le codage est une notion plus large.

Algorithme de César

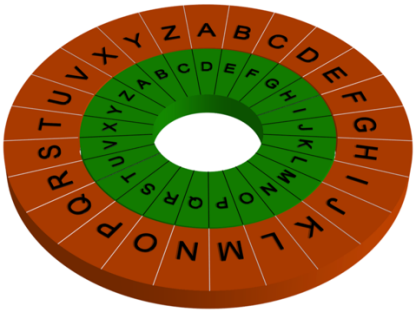
César utilisait une technique simple pour coder ses messages. Cette technique est connue sous le nom de « algorithme de César ». C'est un algorithme de cryptage. C'est un algorithme simple.

Le principe est de décaler l'alphabet dans le message crypté. **La valeur du décalage est la clé de cryptage**.

Ainsi, si la clé vaut 3, A devient D, B devient E, C devient F, …, X devient A, Y devient B, Z devient C.

« CESAR » est crypté par « FHVDU »

On peut représenter les choses ainsi :



L'alphabet rouge est celui du texte en clair (non crypté), l'alphabet vert est celui du texte crypté.

Chiffrage

On peut faire correspondre chaque lettre de l'alphabet à un entier. On définit une bijection « f » d'un ensemble de lettres vers un ensemble d'entiers avec l'image de chaque lettre.

f : {A, B, .., Y, Z} → {0, 1, …, 24, 25}

A → 0, B→ 1, …, Y→24, Z→25

Les lettres sont donc numérotées de 0 à 25 (A=0, B=1, …, Z=25).

Le **chiffrement** consiste donc à :

- Remplacer chaque lettre du texte par l'entier lui correspondant.
- Puis à utiliser la clé pour « chiffrer » la valeur de départ. Dans le cas de l'algorithme de César, il suffit d'ajouter la valeur de la clé (la valeur du décalage).
- Il reste à remplacer le nouveau nombre par la lettre qui lui correspond. A vaut 0. 0 chiffré devient 3. 3 vaut C.

Ensemble des clés

L'ensemble des clés, c'est l'ensemble des valeurs possibles pour la clé.

Dans l'algorithme de César, l'ensemble des clés = {0, 1, …, 24, 25}

Le cardinal de l'ensemble des clés donne la complexité de la clé.

Dans l'algorithme de César, la complexité de la clé vaut 26, ce qui est très peu.

Décodage - Déchiffrement

Pour décoder un texte (pour le déchiffrer) il faut connaître la valeur de la clé (la valeur du décalage).

On peut aussi tester toutes les clés possibles : si le cardinal de l'ensemble des clés est petit, c'est une opération faisable.

De plus, dans le cas de l'algorithme de César, 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.

Exercice

1. **Coder** « bonjour » avec une **clé codage de 6.**
2. **Décoder** «olssv dvysk » avec une **clé de codage de 7**
3. **Chercher « manuellement » à décoder** le texte suivant : « tew wm jegmpi uyi gipe pi gsheki hi giwev »
4. Combien y a-t-il de clés possibles pour l'algorithme de César ?

Exercice en Python (installer anaconda à partir de anaconda.com)

Ecrire un programme qui permet de coder et décoder des textes.

1. Ecrire une fonction coderMessage (message, cle) qui renvoie le message codé.

- Appliquer cette fonction au message décodé de l'exercice 1. Vérifier que vous retrouvez le message codé.
- Comment utiliser cette fonction pour décoder ?

- Appliquer cette fonction au message codé de l'exercice 1. Vérifiez que vous retrouvez le message décodé.

2. Ecrire une fonction frequenceDesLettres (message) qui renvoie un tableau avec la fréquence de chaque lettre de l'alphabet dans le message.

- Affichez le tableau de fréquences des lettres du message codé.
- Affichez le tableau de fréquences des lettres du message décodé.
- Une fonction cleDeDecodage(message) qui renvoie la clé de décodage probable du message (le décalage à appliquer pour décoder).
- Affichez la clé de décodage du message codé.
- Affichez le message décodé à partir du message codé en une seule instruction.

Chiffrement mono alphabétique

On peut complexifier le codage en appliquant aléatoirement une transposition d'une lettre par une autre

Par exemple :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	Q	B	M	X	I	T	E	P	A	L	W	H	S	D	O	Z	K	V	G	R	C	N	Y	J	U

Le nombre de clé est désormais très grand. Il revient au choix d'une bijection de l'ensemble A,B,...,Z vers le même ensemble A,B,...,Z : il y a 26! choix possibles (1*2*3…*25*26).

Exercice

Quel est ce message : « FWXF AFBGF XVG » ?

Défauts

1. La clé est très complexe puisqu'elle contient 26 caractères.
2. Une analyse statistique permet de retrouver facilement le message d'origine : de « craquer » le code ! En effet, les fréquences des lettres sont assez constantes selon les langages :

E	S	A	I	N	T	R	U	L	O	D
14.69%	8.01%	7.54%	7.18%	6.89%	6.88%	6.49%	6.12%	5.63%	5.29%	3.66%

Exercice

Essayer de décoder la phrase suivante :

LHLZ HFQ BC HFFPZ WH YOUPFH MUPZH

Chiffrements plus subtils

- Chiffrement de Vigénere

- Machine Enigma

1 - Division

Division euclidienne = division entière

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 : opérateur « 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

- ***Démonstration***

Faites la démonstration

Modulo : opérateur « mod » ou %

Présentation

L'opération **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

Définition

A % B = R

⇔

A = B*Q + R

avec R < B

Exercices

Sachant que l'on a 96842 = 256 × 375 + 842, déterminer, sans faire la division, le reste de la division du nombre 96842 par chacun des nombres 256 et 375.

Indication : le reste doit être plus petit que le diviseur !

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 $\leq \sqrt{n}$

Méthode pour trouver si un nombre est premier

Pour savoir si un nombre est premier, il faut tester s'il se divise par un nombre premier de la série des nombres premiers en partant de 2. On cherche jusqu'à ce que le diviseur soit plus grand que la racine du nombre à tester.

Exercices

1. Pour chacun des nombres proposés, dire si le nombre est premier.

   On commence par trouver la racine ou une approximation de la racine. Ensuite on passe en revue les diviseurs.

   Si le nombre n'est pas premier, quel est le dernier diviseur testé ?

   Si le nombre est premier, 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. Coder le en python.

Décomposition en facteurs premiers

Définition

Un nombre peut s'écrire comme un produit de puissances de nombres premiers.

Autrement dit :

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

- ***Par exemple :***

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

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.

Exercices

1. Ecrire les nombres entiers suivants sous la forme d'un produit de puissances de nombres premiers. Lister ensuite tous les facteurs : 48, 135, 1617
2. Ecrire l'algorithme qui affiche les diviseurs à partir de la décomposition en nombres premiers. Par exemple, affiche 2, 3, 3, 3 pour 54.

   Faite ensuite une version qui retourne le tableau des résultats : [2, 3, 3, 3]

Tous les diviseurs d'un nombre

Définition

Tout nombre n 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.

- ***Par exemple :***

$54 = 2^1 * 3^3$

Les diviseurs de 54 sont sont de la forme : $2^{ii} * 3^{jj}$ avec $0 \leq ii \leq 1$, $0 \leq jj \leq 3$

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

Exercices

1. Ecrire tous les diviseurs de 135
2. Ecrire l'algorithme qui affiche tous le diviseurs d'un nombre. On se donne la fonction qui retourne le tableau des facteurs premiers et de leurs quantités. Par exemple, pour $54 = 2^1 * 3^3$, on se donne le tableau [(2,1), (3,3)].

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.

Exercices

1. Quel est le PGCD de 315 et 1171. Appliquez la méthode d'Euclide puis la méthode de la soustraction. Idem avec 882 et 540.
2. Ecrivez l'algorithme d'Euclide puis celui de la soustraction en pseudo-code et/ou en python.

Ecrivez une version récursive de l'algorithme d'Euclide.

3 - Congruence

Définition

Formulation

A est congru à B modulo N

⇔

A%N = B%N

⇔

A ≡ B[N]

⇔

A et B sont congrus à N

Exemple :

$7 \equiv 4[3] \Leftrightarrow 7\%3 = 4\%3$

Signification

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

Propriétés principales

A est congru à B modulo N

⇔ A%N = B%N

⇔ A ≡ B[N]

⇔ B ≡ A[N] // commutativité. Démonstration simple par les modulos

⇔ (A-B) % N = 0 // propriété 6 : A-B divisible par N

A%N=R ⇔ A ≡ R[N] et R < N // propriété 7 : congruence et modulo

Propriétés du modulo

Avec A, B non nul, R, n, X, C ∈ N

Propriété 1 : multiplication du dividende et du reste

$$A \% B = R \Leftrightarrow nA \% B = nR$$

- ***Par exemple :***

$11 \% 5 = 1 \Leftrightarrow 3*11 \% 5 = 3*1 \Leftrightarrow 33 \% 5 = 3$

Propriété 2 : augmentation du dividende d'un multiple du diviseur

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

- ***Par exemple :***

$11 \% 5 = 1$

$(11 + 5) \% 5 = 16 \% 5 = 1$

$(11 + 3 * 5) \% 5 = (11 + 15) \% 5 = 26 \% 5 = 1$

Propriété 3 : même addition sur deux dividendes

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

- ***Par exemple :***

$11 \% 5 = 16 \% 5 \Leftrightarrow (11+3) \% 5 = (16+3) \% 5$

Propriété 4 : « distributivité » du diviseur

$$(A + C) \% B = (A \% B + C \% B) \% B$$

- ***Par exemple :***

$(20) \% 5 = (11 + 9) \% 5 = (11 \% 5 + 9 \% 5) \% 5 = (1 + 4) \% 5$

Propriété 5 : modulo du reste

$$\text{si } A \% B = R \text{ alors } A \% B = R \% B = R$$

Par définition, $R < B$ donc $R \% B = R$

Propriété 6 : mixte de 5 et 6 : même addition sur deux dividendes appliquée au reste

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

- ***Par exemple :***

$11 \% 5 = 1 \Leftrightarrow (11 + 7) \% 5 = (1 + 7) \% 5 \Leftrightarrow 18 \% 5 = 8 \% 5$

Propriétés de la congruence

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.

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

- ***Par exemple :***

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

- ***Exercice : démontrer la propriété***

En passant par la définition : A%N = B%N et par une propriété des modulos : augmentation du dividende d'un multiple du diviseur

Propriété 2 : addition et congruence

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

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

- ***Par exemple :***

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

- ***Exercice : démontrer de la propriété***

En passant par la définition : A%N = B%N et par une propriété des modulos : la même addition sur 2 dividendes.

Propriété 3 : multiplication et congruence

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

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

- ***Par exemple :***

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

- ***Exercice : démontrer de la propriété***

En passant par la définition : A%N = B%N et une propriété des modulos : la multiplication du dividende et du reste.

Propriété 4 : puissance et congruence

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

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

- ***Par 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 $AC \equiv BD[N]$

- ***Par exemple :***

$7 \equiv 16[3]$ 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 et et une propriété des modulos : la « distributivité » du dividende

Propriété 6 : A-B divisible par N

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

$A \equiv B[N]$
$\Leftrightarrow$
$A - B \equiv 0[N]$

- ***Par exemple :***

1. $16 \equiv 4[3] \Leftrightarrow (16-4) \% 3 = 0$

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

En passant par la propriété 2 : addition et congruence

Propriété 7 : congruence et modulo

A % N = R ⇔ A ≡ R[N] et R < N

Exemple : si 25 = 3 * 8 + 1 alors 25 ≡ 1[8] et 25 ≡ 1[3]

➢ ***Exercice : démonstration de la propriété***

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

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

➢ ***Cas particulier***

Si A%N=1 alors A ≡ 1[N]

Ce cas est intéressant pour l'inverse du modulo.

Propriété 8 : réduction de la congruence

La réduction d'une congruence consiste à rendre le 2ème terme < modulo.

A ≡ B[N] ⇔ A ≡ B%N[N]

➢ ***Exercice : démonstration de la propriété***

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

Exercices

1. Faites les divisions euclidiennes de 200 et de 900 par 13.
2. Traduisez-les en 2 congruence avec le même modulo (propriété 7).
3. En utilisant les propriétés des congruences, cherchez X dans les formules suivantes et réduisez-le (propriété 8). Vérifiez à chaque fois vos résultats par le calcul.

- 200 + 900 ≡ X [13] (propriété 5).
- 200 x 900 ≡ X [13] (propriété 5).
- 200^2 ≡ X [13] (propriété 4).
- 900^3 ≡ X [13] (propriété 4).
- (200^2 + 900^3)% 13 = X (propriétés 7 et 5).

Inverse du modulo

Définition

Soit a % n,

L'inverse de a % n c'est

le nombre entier $a^{-1} < n$ tel que

$(a* a^{-1}) \% n = 1$

$\Leftrightarrow (a* a^{-1}) \equiv 1[n]$ (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 : **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 le premier multiple de 4 qui, quand on lui ôte 1, est divisible par 9.

Il n'y a pas de méthode de calcul simple pour le trouver !

4-1=3, 8-1=7, 12-1=11, 16-1=15, 20-1=19, 24-1=23, 28-1=27%9 =0.

Réponse : X=7

- ***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)

On cherche le premier multiple de 8 qui, quand on lui ôte 1, est divisible par 27.

8-1=7, 16-1=15, 24-1=23, etc, 17*8=136-1=135 % 27=0.

Réponse : X=17 :

Propriété

(a^{-1} % n) existe

$\Leftrightarrow$

a et n sont premiers entre eux

$\Leftrightarrow$

PGCD (a, n) = 1.

Algorithme : 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 permet de trouver le résultat plus rapidement mais il n'est pas facile à interpréter ! Cf exercice ci-dessous.

- ***Exercices***

1. Coder en python l'algorithme naïf.
2. Coder en python l'algorithme d'Euclide étendu. Utiliser l'algorithme proposé ici.
3. 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 : f(x) = y = ax+b

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 de codage : y ≡ a*x+b[card(E)]

Soit une fonction affine f de E dans E, E étant un intervalle de N débutant en 0.

f(x)=y= (a*x+b) % card(E) avec x, y, a, b appartenant à E.

On peut définir la fonction avec une congruence :

y ≡ a*x+b[card(E)]

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

Décodage : chercher x en fonction de y : $x \equiv a^{-1} *(y - b)\ [card(E)]$

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

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

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 ≡ a*x+b[26] avec x, y, a, b ∈ E

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

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

y ≡ 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} c'est l'inverse de 5 modulo 26. $5*5^{-1} \equiv 1\ [26]$

5^{-1} % 26 = 21 (on trouve le résultat en utilisant un algorithme : 21*5=105, 105 ≡ 1 [26])

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

Exercice

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.
- Afficher la valeur codée de G, puis de S, puis du mot qui était codé par VBUTSB.
- Peut-on utiliser cette fonction pour décoder ?
- Une fonction inverseModulo(a, n) qui calcule l'inverse de a modulo n. On pourra utiliser l'algorithme d'Euclide (ici : https://www.apprendre-en-ligne.net/crypto/rabin/euclide.html) en faisant un copier coller du lien donné dans le cours, ou écrire un algorithme naïf.
- Dans le cas de l'algorithme d'Euclide, pour l'instruction :

```
# q = nombre entier immédiatement inférieur ou égal à no / bo
```

- on pourra écrire :

```
if no%bo==0:
        q=n//bo-1
    else :
        q=no//bo;
```

-
- Une fonction clesDeDecodageAffine(a, b, cles) qui ne renvoie rien mais dont le paramètre « cles » est un tableau (une liste) en sortie qui contient les deux paramètres de la fonction de décodage, autrement dit les 2 clés.

- Utiliser la fonction clesDeDecodageAffine() et la fonction codageAffineMessage() pour décoder VBUTSB