CSI 2501 / Introduction à la théorie des nombres

Si a et b sont des entiers où $a \neq 0$, on dit que a divise b s'il existe un entier c tel que $b=ac$.

$a|b \equiv$ "a divise b" $:\equiv (\exists\, c \in \mathbf{Z}: b=ac)$

On dit que a est un facteur de b et que b est un multiple de a.

On discutera les théorèmes et applications de base de la théorie des nombres.

Applications très importantes:

- Fonctions de hachage,
- Nombres pseudo-aléatoires, Cryptologie,
- Cryptographie à clé publique, Codage avec le
- système RSA, Décodage avec le système RSA.

Introduction à la théorie des nombres

- Propriétés élémentaires:
  - $a \mid 0$
  - Si $a \mid b$ et $a \mid c$, alors $a \mid (b+c)$
  - Si $a \mid b$, alors $a \mid bc$ pour tous les entiers c
  - Si $a \mid b$ et $b \mid c$, alors $a \mid c$

Corollaire: Soient a, b, c des entiers tels que $a \mid b$ et $a \mid c$, alors $a \mid mb + nc$ pour tous les entiers n et m.

Algorithme de Division --- Soit a un entier et soit d un entier positif. Il existe alors deux entiers q et r, avec $0 \leq r < d$, tels que $a = dq+r$. De plus, q et r sont uniques.

On appelle *r* le **reste**, *d* le **diviseur**, *a* le **dividende**, et *q* le **quotient**

Ce n'est vraiment pas un algorithme. C'est plutôt un théorème.

- Si $a = 7$ et $d = 3$, alors $q = 2$ et $r = 1$, puisque $7 = (2)(3) + 1$.
- Si $a = -7$ et $d = 3$, alors $q = -3$ et $r = 2$, puisque $-7 = (-3)(3) + 2$.

Preuve de l'algorithme de Division : (On utilise la propriété du bien ordres de l'ensemble des entiers positifs.)

[Preuve par induction sera faite le chapitre prochain]

Existence: On veut montrer qu'ils existent q et r, tels que $a=dq+r$,

$0 \leq r < d$

On considère l'ensemble S de tous les entiers non-négatifs de la forme $a - dq$, où q est un entier. D'après le principe du bien ordre S a un plus petit élément, $r = a - d\,q_0$.

Prouvons par contradiction que $r < d$. Supposons que $r \geq d$, alors

l'entier $r-d= a-d(q_0+1) \geq 0$ et donc serait dans S. ceci contredit r est le plus entier dans S. Donc, $r < d$ et puisque $0 \leq r$ alors ceci prouve l'existence de $0 \leq r < d$ et de q.

QED

b) **Unicité**

Supposons ∃ q, Q, R $0 \le r$, $R<d$ tels que $a = dq + r$ et $a = dQ + R$.

En toute généralité, on peut supposer que $q \le Q$.

Puisque $dq + r = dQ + R$ alors $d(q-Q) = (R - r)$ et donc d divise $(R-r)$; par conséquent $|d| \le |(R - r)|$ ou $(R - r) = 0$;

Puisque $0 \le r$, $R<d$ alors $-d < R - r < d$ i.e., $|R-r| < d$, donc nécessairement on a $R - r = 0$.

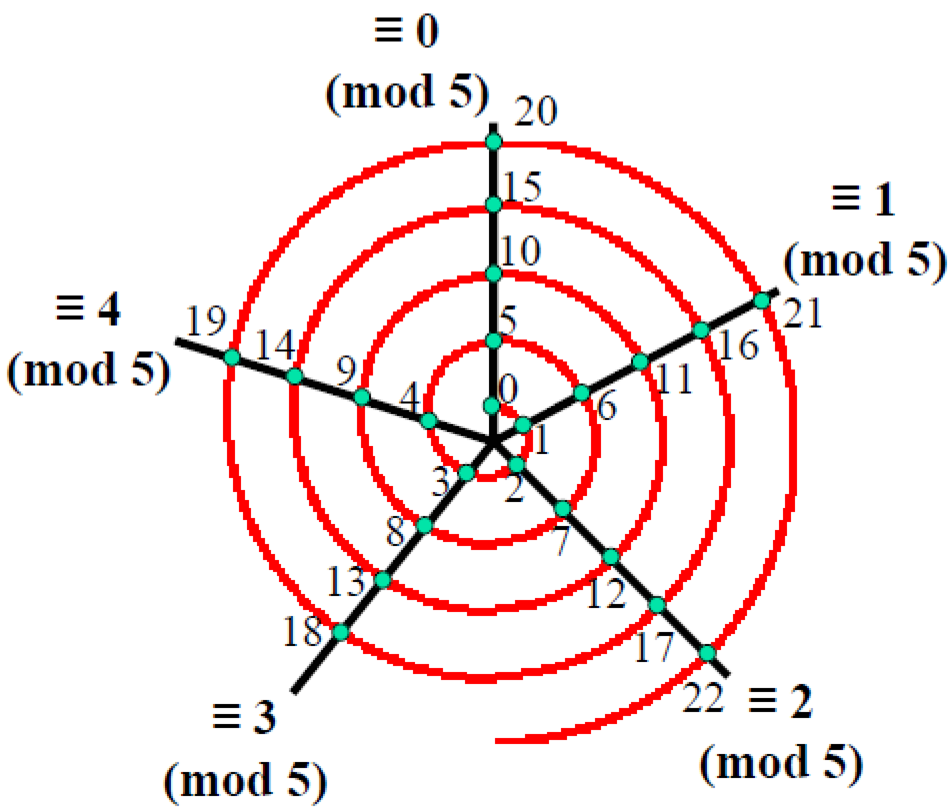
D'ou $R = r$ et donc $dq = d\,Q$ (noter que $d \neq 0$).

Par conséquent $q=Q$, et donc on a prouvé l'unicité.

Arithmétique Modulaire

Soient a et b deux entiers et m un entier positif. Alors **"a est congru à b modulo m"** **si** **m divise a-b**

(dénote: a ≡ b (mod m) ; **a mod m = b mod m)**



6 divise 17-5,
17 est congru à 5 modulo 6,
17 ≡ 5 (mod 6)

Classes de Congruence modulo 5

Manipulations algébriques des congruences

"a est congru à b modulo m" si m divise a-b

Théorème: Soit m un entier positif. Les entiers a et b sont congrus modulo m si est seulement s'il existe un entier k tel que $a = b + km$

Théorème: Soit m un entier positif. Si $a \equiv b \pmod{m}$ et $c \equiv d \pmod{m}$, alors $a+c \equiv (b+d) \pmod{m}$ et $ac \equiv bd \pmod{m}$

- On peut multiplier des deux cotés une congruence par un entier.
- On peut additionner un entier aux deux cotés d'une congruence.
- Diviser des deux cotés une congruence par un entier n'est pas toujours une opération permise.

Manipulations algébriques des congruences

Exemples:

Puisque 7 $\equiv$ 2 (mod 5) et 11 $\equiv$ 1 (mod 5), alors d'après le théorème ci-dessus:

$$18 = 7 + 11 \equiv 2 + 1 = 3 \pmod 5$$

$$77 = 7 * 11 \equiv 2 * 1 = 2 \pmod 5$$

On a $14 \equiv 8 \pmod 6$ mais si on divise des deux cotés par 2 ($14/2 = 7$ and $8/2 = 4$) alors on obtient $7 \not\equiv 4 \pmod 6$.

Arithmétique Modulo *m*

Définitions:

- Soit $\mathbf{Z}_m$ l'ensemble des entiers plus petit que m: $\{0, 1, \ldots, m-1\}$
- L' opération $+_m$ *(addition modulo m)* est défini par $a +_m b = (a + b)$ **mod** m.
- L' opération $\cdot_m$ *(multiplication modulo m)* est défini par $a \cdot_m b = (a + b)$ **mod** m.

Exemple: Calculer $7 +_{11} 9$ and $7 \cdot_{11} 9$.

Solution: a partir de la définition ci-dessus:

- $7 +_{11} 9 = (7 + 9)\ \mathbf{mod}\ 11 = 16\ \mathbf{mod}\ 11 = 5$
- $7 \cdot_{11} 9 = (7 \cdot 9)\ \mathbf{mod}\ 11 = 63\ \mathbf{mod}\ 11 = 8$

Arithmétique Modulo *m*

- *Inverses pour l'addition* :
  - Si $a \neq 0$ est dans $\mathbf{Z}_m$, alors $m - a$ est l'inverse de a modulo m. L'entier 0 est son propre inverse.

    $a +_m (m - a) = 0$ et $0 +_m 0 = 0$
- *Inverses pour la multiplication n'existe pas tout le temps. Par exemple 2 n'a pas d'inverse modulo 6.*
- *Identités:*

  Si a est dans $\mathbf{Z}_m$, alors $a +_m 0 = a$ et $a \cdot_m 1 = a$.

Fonctions de hachage

Comment affecter des adresses mémoires a des registres pour être capable de les récupérer rapidement?

Solution a ce problème $\rightarrow$ une bonne fonction de hachage.

On identifie les registres a l'aide d'une clé, qui désigne d'une façon unique le registre correspondent.

Fonctions de hachage

Un fonction de hachage h: A→B d'un ensemble A "l'ensemble des clés" sur un ensemble "plus petit" B "adresses mémoires" (i.e., $|A|\geq|B|$).

Une fonction de hachage efficace doit avoir les propriétés suivantes:

- La fonction est surjective.
- Efficace a calculer.
- La cardinalité des images inverses des éléments de B doivent être de tailles semblables.

  $\forall b_1,b_2\in B: |h^{-1}(b_1)| \approx |h^{-1}(b_2)|$

  Donc les éléments de B seront générés avec une probabilité presque uniforme.
- Idéalement, la fonction doit apparaitre le plus possible aléatoire, de façon a ce que des éléments similaires dans A n'auront probablement pas les mêmes images ou des images similaires dans B.

Fonctions de hachage

Qu'est ce qui est important pour les fonctions de hachage?

- Calcul rapide et efficace.
- Pour que chaque message soit haché.
- Pour prévenir qu'un message soit remplacé par un autre avec la même valeur de hachage.

Important: Fonction de hachage soit sécurisé (en terme cryptographique):

- Etant donne un élément *b∈B*, trouver un *a∈A* tel que h(a)=b doit avoir une complexité en moyenne de $\Omega(|B|^c)$ pour un certain $c>0$.

  Pour s'assurer que ca prendra un temps exponentiel en terme de la longueur de l'ID, si on veut créer un document forgé avec le même ID.

$A = \{a \in \mathbf{N} \mid a < a_{lim}\}$, $B = \{b \in \mathbf{N} \mid b < b_{lim}\}$

- Une fonction de hachage simple de A sur B (ou $a_{lim} \geq b_{lim}$) est $h(a) = a \bmod b_{lim}$.

- Elle a les propriétés de base requises
- Pas très aléatoires.
- Pas sécurisée: assez simple de générer a dont l'image est b

On sait que pour tout $n \in \mathbf{N}$, $h(b + n\, b_{lim}) = b$.

Systèmes de signature digitales

- Plusieurs systèmes de signature digitales utilise des fonctions de hachages cryptographiquement sécurisées (mais publiques) h et qui envoie des longs documents arbitraires a un mot de longueur fixe (e.g., 1,024-bit) “fingerprint” .
- Procédure de signature des documents: Etant donne un document *a* a signer, calculer rapidement son hachage $b=h(a)$.

    –Calculer une certaine fonction $c = f(b)$ que seul le signataire connait. (En général c’est une étape complexe et lente, qu’on ne veut pas appliquer au document en entier.)

    –Délivrer le document original avec la signature *c*.

- Procédure de vérification de la signature : Etant donné un document *a* et une signature *c*, calculer rapidement $b = h(a)$.

    –Calculer $b' = f^{-1}(c)$. (Possible si l’inverse f^{-1} de *f*, est publique (mais pas *f*).)

    –Comparer *b* et *b'*; s’ils ont la même valeur alors la signature est valable.

Si h n’est pas sécurisée cryptographiquement , alors on peut facilement forgé un document a’ différent de a mais qui la même valeur b par la fonction de hachage et puis attacher la signature de quelqu’un a un autre document différent de celui qu’il a signé!

Nombres pseudo-aléatoires

Les ordinateurs ne peuvent pas généraliser des vrai nombres aléatoires – on les appelles nombres pseudo-aléatoires!

- **Méthode linéaires de congruence** (Algorithme assez courant pour générer des nombres pseudo-aléatoires.)

  Choisir 4 entiers

  - **Semence** x_0: valeur de départ
  - **Module** m: valeur maximale possible
  - **Multiplicateur** a: tel que $2 \leq a < m$
  - **Incrément** c: entre 0 et m
- Pour générer une suite de nombres pseudo- aléatoires,

$\{x_n \mid 0 \leq x_n < m\}$, appliquer la formula $x_{n+1} = (ax_n + c) \bmod m$

Nombres pseudo-aléatoires

Formule: $x_{n+1} = (ax_n + c) \bmod m$

Soient $x_0 = 3$, $m = 9$, $a = 7$, et $c = 4$

- $x1 = 7x_0+4 = 7*3+4 = 25 \bmod 9 = 7$
- $x2 = 7x_1+4 = 7*7+4 = 53 \bmod 9 = 8$
- $x3 = 7x_2+4 = 7*8+4 = 60 \bmod 9 = 6$
- $x4 = 7x_3+4 = 7*6+4 = 46 \bmod 9 = 1$
- $x5 = 7x_4+4 = 7*1+4 = 46 \bmod 9 = 2$
- $x6 = 7x_5+4 = 7*2+4 = 46 \bmod 9 = 0$
- $x7 = 7x_6+4 = 7*0+4 = 46 \bmod 9 = 4$
- $x8 = 7x_7+4 = 7*4+4 = 46 \bmod 9 = 5$

Nombres pseudo-aléatoires

Formule: $x_{n+1} = (ax_n + c) \bmod m$

Soient $x_0 = 3$, $m = 9$, $a = 7$, et $c = 4$

- La suite générée:

  3, 7, 8, 6, 1, 2, 0, 4, 5, 3 , 7, 8, 6, 1, 2, 0, 4, 5, 3
- C'est cyclique!
- Génère tous les nombres possibles avant de répéter les mêmes nombres
- L'algorithme le plus connue utilise $m = 2^{32}-1$
  - To dois choisir 4 milliards nombres avant de répéter
  - Multiplicateur $7^5 = 16{,}807$ et incrément c=0 (générateur purement multiplicatif)

Cryptologie (messages secrets)

Méthode de codage de César: Jules César fut l'un des premiers a utiliser la cryptologie.

- On remplace chaque lettre par un entier entre 0 et 25.
- Une fonction f de codage qui affecte a p, $f(p)=(p+3)\bmod 26$ où p est une lettre (0 est A, 1 est B, 25 est Z, etc.)
- Décodage: $f^{-1}(p) = (p-3) \bmod 26$

Codage par substitution

Codage de César

Coder “on attaque”

- Traduire en nombres: o = 14, n = 13, etc.
- Suite complète: 14, 13, 0, 19, 19, 0, 16, 8, 20, 4
- Appliquer le codage a chaque nombre: $f(6) = 9$, $f(14) = 17$, etc.
- Suite complète: 17, 16, 3, 22, 22, 3, 19, 11, 23, 7
- Convertir les nombres en lettres 17 = r, 16 = q, etc.
- Suite complète: rq dxxdtyh

Décoder “rq dxxdtyh”

- Traduire en nombres: r = 17, q = 16, etc.
- Suite complète: 17, 16, 3, 22, 22, 3, 19, 11, 23, 7
- Appliquer le décodage a chaque nombre: $f^{-1}(17) = 14$, $f^{-1}(16) = 13$,
- Suite complète: 14, 13, 0, 19, 19, 0, 16, 8, 20, 4
- Convertir les nombres en lettres o = 14, n = 13, etc.
- Suite complète: “on attaque”

Codage Rot13

Le codage de César avec une translation de 13. La même fonction est utilisée pour coder et pour décoder.

(Rot13: rotation de 13)

Exemple:

Hello World | rot13

Uryyb Jbeyq

Uryyb Jbeyq | rot13

Hello World

Théorème fondamental de l'arithmétique

Un entier positif p plus grand que 1 est appelé nombre premier si les seuls facteurs positifs de p sont 1 et p.

Un entier positif qui est plus grand que 1 est qui n'est pas premier est appelé nombre composé. (1 n'est pas premier et il n'est pas composé, il est dans une classe a part)

Théorème fondamental de l'arithmétique: Tout entier positif peut s'écrire comme le produit des nombres premiers et de façon unique (l'ordre des facteurs n'étant pas pris en considération dans l'unicité).

Les nombres premiers sont les structures de base des entiers.

Preuve du Théorème fondamental de l'arithmétique :

(On utilise une preuve par Induction généralisée qui sera faite le chapitre prochain]

On démontre P(n): si n est un entier plus grand que 1 alors n s'écrit comme le produit des nombres premiers.

- Etape de base – P(2): 2 s'écrit comme produit de lui-même.
- Etape Inductive - Supposons que P(j) est vrai pour $\forall\ 2 \leq j \leq k$, j entier. Prouvons que P(k+1) est vrai.

a) Si k+1 est premier alors il est le produit de lui-même et donc P(k+1) est vrai;

b) Si k+1 est composé alors il s'écrit comme produit de deux entiers positifs a et b, avec $2 \leq a \leq b \leq k+1$. D' après l'hypothèse inductive, a et b s'écrivent comme produits des nombres premiers et par conséquent k+1 aussi.

Il reste a prouver l'unicité de la décomposition: on a besoin de plus de connaissances …

Théorème fondamental de l'arithmétique

Théorème: Si n est un entier composé alors n admet un diviseur premier plus petit que, ou égal a $\sqrt{n}$.

Preuve: Puisque n est composé alors n est divisible par un entier a tel que $1<a<n$. Donc $n = ab$, où a et b sont des entiers positifs plus grand que 1. Nécessairement $a\leq\sqrt{n}$ ou $b\leq\sqrt{n}$ (autrement, $ab > \sqrt{n}*\sqrt{n} > n$. Contradiction.)

Donc, n a un diviseur plus petit que, ou égal a $\sqrt{n}$. D'après le théorème fondamental d'arithmétique, ce diviseur admet un facteur premier et donc n admet un facteur premier plus petit que, ou égal a $\sqrt{n}$.

Exemple: prouver que 113 est premier.

Preuve: Les seuls facteurs premiers plus petit que, ou égal a $\sqrt{113} = 10.63$ sont 2, 3, 5, and 7. Aucun de ces entiers ne divise 113. D'après le théorème fondamental d'arithmétique, 113 est premier.

Nombres de Mersenne

Nombre de Mersenne: Nombre sous la forme 2^n-1

Premier de Mersenne: Nombre premier sous la forme 2^p-1, ou p est aussi premier.

2^5-1 = 31 est un premier de Mersenne. Mais 2^{11}-1 = 2047 n'est pas premier(23*89)

Si M est un nombre premier de Mersenne alors M(M+1)/2 est un nombre parfait (un nombre est parfait s'il est égal a la somme de ses diviseurs.)

2^3-1 = 7 est un premier de Mersenne, donc 7*8/2 = 28 est un nombre parfait (28 = 1+2+4+7+14)

2^5-1 = 31 est un premier de Mersenne, donc 31*32/2 = 496 est un nombre parfait (496 = 2*2*2*2*31, 1+2+4+8+16+31+62+124+248 = 496)

Les plus grands nombres premiers qu'on connait sont des nombres premiers de Mersenne. Puisque, 2^p-1 grandit très rapidement ceci est un test très efficace de primalité– test de Lucas-Lehmer – des premiers de Mersenne.