

Résumé

Ce devoir est une synthèse de nos travaux depuis plusieurs mois sur la reconstitution du théorème des restes chinois. Grace au soutiens de notre professeur MR ZEMOR, nous avons appris a reconstruire le théorème des restes chinois en présence d'erreur ; Dans un premier temps, nous avons pris connaissance de ce théorème, puis dans un second temps, nous l'avons corrigé grace à plusieurs méthode, tel que du brut force ou une méthode de fractions continues.

Table des matières

Nous tenons tout d'abord à remercier Mr CASTAGNOS Guillaume d'avoir été le professeur principale de cette matière et, en ayant bien expliqué le fonctionnement pour partir sur de bonne base. Nous remercions par la suite Mm ZEMOR Gilles pour nous avoir suivit, aider et tutoré durant ce semestre.

nous remercions également nos familles et nos proches, pour leurs aide et leur soutiens.

1 théorème des restes

1.1 théorème des restes chinois et son histoire

Le théorème des restes chinois viens à la base d'un livre mathématique chinois de Qin JUSHIO publié en 1247. Cependant, on avait déjà découvert ce théorème au part avant dans un livre de Sun ZI au 3° siècle. Le théorème consiste en : On pose n_1, \dots, n_k des entiers premiers 2 à 2. Pour tout a_1, \dots, a_k , il existe un entier x tel que :

$$x \equiv a_1 \pmod{n_1}$$

$$x \equiv a_2 \pmod{n_2}$$

$$x \equiv a_k \pmod{n_k}$$

Nous pouvons démontrer ce théorème de la façon suivante :

Pour illustrer ce théorème, nous allons donner un exemple, mais pas n'importe quelle exemple, celui dont Sun ZI a proposé une solution :

soient des objets, prenons des bonbons, si on les repartis pour 3 enfants, il en reste 2, si on les répartis pour les 3 enfants et leurs parents (soit 5 personnes), il en reste 3. enfin si on partage ces bonbons avec également les 2 cousins, (soit 7 personnes) il en reste 2. On a donc :

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

La question que l'on se pose à présent est combien y a t'il de bonbons ?

Grace au théorème des restes chinois, on peut trouver la réponse. Enfin, nous avons plusieurs réponse, c'est à dire que tout les nombres congru a x modulo n sont des bonnes réponses. Avant de répondre à ce problème, il faut d'abord voir l'algorithme.

1.2 notre algorithme

L'algorithme du théorème des restes chinois étant un peu lourd, nous allons montrer l'algorithme comme suivant avant de voir plus en détail les algos en python. On a ici :

Soit n_i le i -ème termes de la liste des modulus, et on note

$$N_i = \frac{n}{n_i} = n_1 n_2 \dots n_i - 1 n_i + 1 \dots n_k$$

on a donc N_i et n_i qui sont premier entre eux.

Il faut alors faire l'algorithme d'euclide étendu sur N_i et n_i , se qui nous donne :

$$1 = N_i u_i + n_i v_i \text{ où } u_i, v_i \in \mathbb{Z}$$

On pose donc $e_i = u_i N_i$ avec $e_i \equiv 1 \pmod{n_i}$ et $e_i \equiv 0 \pmod{n_j}$ avec $i \neq j$

On trouve donc une solution qui est $x = \sum_{i=1}^k a_i e_i$

Nous pouvons à présent répondre à l'exemple précédent. Nous avons :

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x_k \equiv 2 \pmod{7}$$

On a $N_1 = 5 \times 7 = 35$, $N_2 = 3 \times 7 = 21$, et $N_3 = 3 \times 5 = 15$

On fait l'algorithme d'euclide sur N_1 et n_1 qui donne $-3 \times 23 + 2 \times 35 \times 1 = 1$ et aussi $e_1 = 2 \times 35$

Idem sur N_2 et n_2 qui donne $21 \times 1 - 5 \times 4 = 1$, et donc $e_2 = 21$

Enfin, on a $15 \times 1 - 7 \times 2 = 1$ avec $e_3 = 15$

Le résultat final est donc $x = 2 \times 35 + 3 \times 21 + 2 \times 15 = 233$. Nous avons, comme dit précédement plusieurs résultat, qui sont les entiers congrus à 233 modulo 105

Dans un premier temps, il a fallu faire l'algorithmes d'Euclide étendu, qui prend en paramètres a et b, deux entiers, et qui renvoie l'inverse de a modulo b.

2 Brute force

Il existe plusieurs algorithmes de brut force qui permette de résoudre les erreurs tel que essayer tout les cas possibles. Mais cette algorithme est trop lourd à écrire. Nous avons donc fait des algorithmes moins lourds. Nous vous présentons 2 algorithmes dans cette section.

2.1 Notre premier algo version naïve

Tout d'abord, nous avons fait une fonction qui nous montre si il y a une erreur ou pas dans nos listes, car avant de vouloir trouver les erreurs, il faut savoir si il y en a une. Nous avons donc fait un petit algorithme qui enlève, chacun leurs tours, un élément des listes; c'est à dire qu'on retire le premier élément des 2 listes, puis on retire que le deuxième élément, puis le troisième et ainsi de suite. Nous appliquons alors le théorème des restes chinois à chaque itération, et on met les résultats dans une liste auxiliaire. Si il n'y a pas d'erreur, lorsqu'on retourne cette liste auxiliaire, on devrait avoir la même valeur partout, si les valeurs de la liste sont différentes, nous avons une erreur.

Soit $P = [p_1, p_2, \dots, p_k]$ et $N = [n_1, n_2, \dots, n_k]$

On note $P_i = [p_1, \dots, p_i - 1, p_i + 1, p_k]$ et $N_i = [n_1, \dots, n_i - 1, n_i + 1, \dots, n_k]$

On note x_i le résultat théorème des restes chinois appliqué à P_i et N_i

Puis on retourne $L = [x_1, \dots, x_k]$

Si $x_1 = x_2 = \dots = x_k$ alors il n'y a pas d'erreur.

Voici un petit exemple pour bien comprendre cette fonction.

on pose $P=[2,3,5,7,11,13,17]$ et $N=[1, 1, 1, 5, 6, 9, 10]$

si on applique le théorème des restes chinois à ces 2 listes, on trouve 510510.

Pour vérifier qu'il n'y a pas d'erreur, on applique le théorème à $P1=[3,5,7,11,13,17]$

et $N1=[1, 1, 5, 6, 9, 10]$, puis à $P2=[2,5,7,11,13,17]$ et $N2=[1, 1, 5, 6, 9, 10]$

jusqu'à $P7=[2,3,5,7,11,13]$ et $N7=[1, 1, 1, 5, 6, 9]$

Comme il n'y a pas d'erreur, la fonction va retourner [61, 61, 61, 61, 61, 61, 61]

à présent, nous mettons une erreur dans les listes, on a $P=[2,3,5,7,11,13,17]$ et $N=[1,1,1,5,8,10]$. En appliquant la fonction, il sera retourner une liste [...], où l'on remarque bien que chaque élément est différent, donc nous avons au plus une erreur. Dans cette fonction, on parcourt n fois la liste de n éléments donc nous sommes en $O(n^2)$. Mais on fait n fois l'algorithme des restes chinois donc on est en $O()$

Pour créer notre première fonction de correction, nous nous sommes inspirées de la fonction ci dessus. On sait dit que si on enlève 2 éléments des listes, nous pourrions trouver l'erreur. On s'explique; On a vu que si on enlevait 1 élément des listes, le théorème des restes chinois marchait encore,

et si on enlève 2 éléments, il marche encore. Donc, en enlevant le premier élément et tout les autres chacun leurs tours, et en appliquant le théorème à chaque fois, on retourne une liste. Puis, on réitère en enlevant le deuxième élément, et les autres chacun leurs tours, en re-applicant les restes chinois à chaque fois, en mettant les résultats dans une nouvelle liste. Ainsi de suite jusqu'au dernier élément. Cette fonction retourne une liste de liste. Donc la liste où se trouve le même nombre partout, c'est que l'erreur est à la même position. faisons plutôt exemple.

Soit $P = [p_1, p_2, \dots, p_k]$ et $N = [n_1, n_2, \dots, n_k]$

On note $P_{ij} = [p_1, \dots, p_i - 1, p_i + 1, \dots, p_j - 1, p_j + 1, \dots, p_k]$ et $N_{ij} = [n_1, \dots, n_i - 1, n_i + 1, \dots, n_j - 1, n_j + 1, \dots, n_k]$

On note x_{ij} le résultat du théorème des restes chinois appliqués à P_{ij} et N_{ij}

Et on retourne $L = [[x_{11}, x_{12}, \dots, x_{1k}], [x_{21}, x_{22}, \dots, x_{2k}], \dots, [x_{k1}, x_{k2}, \dots, x_{kk}]]$

On a retourné une liste de liste. Si la k-ième sous-liste retourne toujours le même nombre alors cela signifie qu'il y a une erreur sur le k-ième élément.

Cette fonction est très performante comme vous avez pu le voir sur les exemples, cependant il y a quelque défaut. Premièrement, cette fonction ne marche que avec une erreur, on peut facilement changer la fonction pour trouver n erreur juste en rajoutant des boucles FOR. Deuxièmement, nous avons un problème de borne. Pour trouver la borne, il suffit de multiplier par les 2 derniers nombre premier de la liste.

Par exemple pour le nombre 30030, la borne est 11x13 soit 210.

Enfin, nous avons un problème de .. soit $O(n^2)$ et à chaque fois on fait le théorème des restes chinois.

2.2 brute force de hamming

3 Les fractions continues

3.1 algo

3.2 jesaispas

```
1  $xs0 \leftarrow 1$  ;  
2  $xs1 \leftarrow 0$  ;  
3  $ys0 \leftarrow 0$  ;  
4  $ys1 \leftarrow 1$  ;  
5  $s \leftarrow 1$  ;  
6  $d \leftarrow b$  ;  
7 while  $b \neq 0$  do  
8    $(q, r) \leftarrow \text{divmod}(a, b)$  ;  
9    $(a, b) \leftarrow (b, r)$  ;  
10   $(x, y) \leftarrow (xs1, ys1)$  ;  
11   $(rs1, ys1) \leftarrow (q \times s1 + xs0, q \times ys1 + ys0)$  ;  
12   $(xs0, ys0) \leftarrow (x, y)$  ;  
13   $s \leftarrow -s$  ;  
14 end  
15 return  $s \times xs0 + ((1 - s) \div 2) \times d$ 
```