Preuve et Analyse des Algorithmes

14:00–16:00
aucun document autorisé

4 janvier 2016

1 Stabilité d'un algorithme de tri

Un algorithme de tri qui compare et déplace les éléments d'un tableau est dit stable s'il ne change pas l'ordre relatif de deux éléments égaux pour la clé de tri considérée. Autrement dit, si $t[i] = t[j]$ pour deux indices $i < j$ à l'entrée de l'algorithme de tri, alors la position finale de l'élément $t[i]$ est inférieure à celle de $t[j]$.

1. Donner un exemple d'algorithme de tri stable. Le tri par insertion, le tri fusion à condition de fusionner correctement sont stables. Le tri par sélection n'est pas stable.
2. Compléter la ligne 10 de la coupure de Lomuto. Il faut ajouter la permutation `swap( x[left], x[i] )`
3. Etudier la stabilité du tri basé sur Lomuto. Le tri rapide basé sur la coupure de Lomuto n'est pas stable, par exemple, si $a = b < t$, elle produit :

t	a	b		
i	j			$x[j] < t$
	ij			
t	a	b		
	i	j		$x[j] < t$
		ij		
t	a	b		
		i	j	
b	a	t		

```
1  procedure split(x, left, right, i)
2  T := x[left];
3  i := left;
4  for j := left + 1 to right do
5    if x[j] < T  then
6   i := i + 1
7   swap(x[i], x[j])
8    fi
9  done
10 // ???
11 end. {split}
```

Lomuto

```
int primitif ( int m)
{
int x, y, r;
for( x= 2; x < m; x++){
    r = 0; y=1;
    do {
        r++; y = (x*y) % m;
    } while ( y != 1 );
    if ( r = (m-1) ) return x;
}
}
```

2 Résidu primitif

Soit m un nombre premier. On appelle résidu modulo m tout entier naturel inférieur à m. Pour tout entier naturel z, il existe un et un seul résidu, que nous noterons $[z]_m$, tel que

$$\exists q \in \mathbf{N}, \quad z = qm + [z]_m \tag{1}$$

On rappelle que la multiplication modulaire $(x, y) \mapsto x \otimes y := [xy]_m$ définit un opérateur binaire associatif sur les résidus modulo m. Un résidu g est dit primitif modulo m si pour tout résidu non nul y modulo m, il existe un entier k tel que $y = [g^k]_m$, on dit alors que k est le logarithme à base g de y. Un résultat classique de la théorie des nombres, que nous admettrons, affirme que tout nombre premier m admet au moins un élément primitif.

1. Quelle notion mathématique permet d'affirmer l'existence et l'unicité du résidu décrit dans la relation (1). La notion de division euclidienne !
2. Quels sont les résidus primitifs modulo 7 ?

	0	1	2	3	4	5	k
2	1	2	4	1	2	4	
3	1	3	2	6	4	5	primitif
4	1	4	2	1	4	2	
5	1	5	4	6	3	2	primitif
6	1	6	1	6	1	6	
x							$[x^k]_7$

3. Montrer que si x est un résidu quelconque alors il existe deux entiers $i < j$ tel que $[x^i]_m = [x^j]_m$, sans que m ne soit nécessairement premier. L'ensemble des entiers est infini, celui des résidus est fini.
4. Montrer que si x est un résidu non nul (m premier) alors il existe un entier $k > 0$ tel que $[x^k]_m = 1$. En effet, si $j > i$ alors en notant t l'inverse de x modulo m et en multipliant par t^i,

$$[x^i]_m = [x^j]_m \quad \Longrightarrow 1 = [x^0]_m = [x^{j-i}]_m$$

5. Montrer que g est primitif si et seulement si

$$\forall k, \qquad 0 < k < m-1 \quad \Longrightarrow \quad [g^k]_m \neq 1.$$

   L'application $k \mapsto [x^k]_m \neq 1$ est une bijection si et seulement si x est primitif.
6. Ecrire une fonction `primitif(int m)` qui renvoie le plus petit résidu primitif modulo m.
7. Ecrire une fonction `int dl (int y, int g, int m)` qui détermine le logarithme de y en base g sachant que g est primitif modulo m.
8. Préciser le temps de calcul moyen. Le temps moyen est $\frac{m-1}{2}$.

```
1 int dl( int y, int g, int m)
2 {
3 int r, t;
4 for( r=0, t=1 ; y != t, r++, t= (t*g) % m ) ; // nothing
5 return r;
6 }
```

```
1 char val( ullong z )
2 {
3 int r = 0;
4 while ( 0 == z & 1 ) {
5         r++;
6         z>>=1;
7 }
8 return r;
9 }
```

3 Logarithme des mots de poids 1

Un mot binaire de poids 1, s'écrit 2^v pour un certain v. Dans cet exercice, on s'intéresse aux fonctions écrites en Langage C qui déterminent l'entier v pour des entiers de 64 bits mais de poids 1.

1. Donner un exemple de contexte d'utilisation. La gestion des sous-ensembles de $\{0, 1, \ldots, 63\}$.
2. Compléter la fonction `char val(ullong z)` en privilégiant l'utilisation des opérateurs bit-à-bit du Langage C.
3. Le type de retour de la fonction `val` est-il correct ? Le logarithme est inférieur à 64, le choix du type `char` est correct.
4. Préciser les temps de calcul. Cas favorable 1 étape, pire des cas en 64 étapes et 32 étapes en moyenne.
5. Ecrire une fonction `int log(ullong z)` plus efficace. On peut utiliser la dichotomie !
6. Préciser le temps de calcul. 6 étapes.
7. Décrire une solution basée sur le fait que 2 est primitif modulo 67. Il suffit de précalculer une table T telle que $T[(2^v) \bmod 67] = v$, le logarithme de z sera donné par $T[z \bmod 67]$.