Preuve et Analyse des Algorithmes

Module I31, Licence Informatique, USTV.

Juin 2011

↯ *Le sujet est composé 5 exercices à traiter en moins de 0x5A minutes. Aucun document autorisé. Vous êtes invités à remettre une copie claire, concise, sans rature ni surcharge. Il est par ailleurs inutile de recopier l'énoncé... La note finale tiendra compte de la présentation générale de la copie.*

1 Somme

On rappelle que si t désigne un nombre réel alors e^{it} désigne le nombre complexe $\cos(t) + i\sin(t)$. Soit n un entier non nul. On rappelle la formule (De Moivre) $\left(e^{it}\right)^n = e^{int}$. On note z le nombre complexe $e^{2i\pi/n}$.

1. Que vaut z et que vaut la somme
$$1 + z + z^2 + z^3 =$$
dans le cas $n = 4$?
2. En général, que vaut :
$$z^n =$$
3. et que vaut la somme :
$$1 + z + \cdots + z^{n-1} =$$

2 Notations asymptotiques

1. Vrai ou faux ? Justifier.
$$1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = O(n)$$
2. Montrer que
$$1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n-1} \sim \ln(n)$$

3 Temps de calcul

```
1  Algorithme E( n : entier)
2  variable  i, r : indice;
3       t : table de n booleen;
4  debut
5    r := 0;
6    i := 2;
7    t := [1,1,...,1];
8    tantque ( i < n )
9         si ( t[i] = 1 )
10             alors
11             j := i;
12             tantque ( j < n )
13               t[ j ] := 0;
14               j := j +  i;
15             ftq
16        r:= r+1;
17        fsi
18        i := i + 1
19   ftq
20   retourner r;
21 fin
```

1. On note $t(i,n)$ le nombre d'itérations de la boucle interne (lignes 13,14) en fonction de i et de n. Montrer que
$$t(i,n) \leq \frac{n}{i}$$
2. En déduire que le nombre total $I(n)$ de ces itérations vérifie
$$I(n) = \mathrm{O}\big(n\log(n)\big).$$
3. Pour quelles valeurs de $i < n$,
$$t(i,n) = 0?$$
4. Estimer le temps de calcul de l'algorithme.
5. Que fait cet algorithme ?

4 Grands entiers

On représente les grands entiers comme dans les séances de travaux pratiques :

```
typedef unsigned int uint;
uint taille = 1024;
uint base   = 10;
typedef uint * nombre;
```

1. Ecrire une fonction en langage C
`nombre iton( uint v)`
qui retourne le nombre ayant pour valeur v.
2. Ecrire une fonction en langage C
`uint ntoi( nombre z, uint p)`
qui retourne la valeur du nombre z modulo p.
3. Ecrire une fonction efficace
`uint parite( nombre z)`
qui retourne la parité de z. La fonction tiendra compte de la parité de la base.

5 Invention

Soit n un entier. On note X_n l'ensemble $\{0, 1, \ldots, n-1\}$, $\mathcal{R}$ une relation binaire sur X_n. On rappelle qu'une relation d'équivalence est une relation : réflexive, symétrique et transitive. Si $\mathcal{R}$ désigne une relation d'équivalence, la classe de x désigne l'ensemble des éléments en relation avec x.

1. Comment peut-on utiliser la structure de tableau pour modéliser une relation binaire sur l'ensemble X_n ?

2. Préciser la délaration de type en langage C pour définir un type `relation`.

3. Ecrire une fonction `uint test(relation R)` qui renvoie vrai si R est une relation d'équivalence.

4. Ecrire une fonction `uint classe(relation R, uint x)` qui compte le nombre d'éléments en relation avec x par R.

5. Préciser le temps de calcul.

6. Ecrire une fonction `uint quotient( relation R)` qui calcule le nombre de classes d'équivalences suivant R. *Remarque : il est possible modifier l'algorithme E pour répondre à cette question.*

7. Préciser le temps de calcul.