Université de Nice-Sophia Antipolis — Licence Informatique 2
Outils Formels pour l'Informatique — 2014–2015

Contrôle continu Lundi 2 Novembre

Durée : 1h30

Toutes les réponses doivent être justifiées. Le correcteur attachera de l'importance à la qualité de rédaction. Les documents, calculatrices ou téléphones portables ne sont pas autorisés. Le barême, sur 40, est donné à titre indicatif.

1 Complexité (4 points)

Calculer les complexités des algorithmes suivants

1.1 Algorithme 1

Algorithme 1

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i;  j++) {
    x = x + 1; }
  }
```

Le nombre d'opérations élémentaires dans l'algorithme 1 se calcule par $\Sigma_{i=0}^{n-1}\Sigma_{j=0}^{i-1} = \Sigma_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ et l'algorithme 1 est donc de complexité $O(n^2)$.

. . . . . .

1.2 Algorithme 2

Algorithme 2

```
for (int i = 0; i < n; i++) {
    for (int j = i; j < n-i;  j++) {
    x = x + 1; }
  }
```

On considère tout d'abord le cas où $n = 2k+1$ est impair, pour lequel le nombre d'opérations élémentaires est $\Sigma_{i=0}^{(n-1)/2}\Sigma_{j=i}^{n-i-1} 1 = \Sigma_{i=0}^{k}(2k-2i)1 = k(k+1) = (n-1)/2((n-1)/2+1)$ soit encore une complexité quadratique. Le cas pair se traite de la même manière.

. . . . . .

2 Ordre de grandeur (4 points)

Donner l'ordre de grandeur des expressions suivantes :

1. $f(n) = \frac{2}{3}n^2 - 3n + 1$
2. $g(n) = \frac{n^2\log(n) - n + \log(n)}{n-1}$

On écrit $f(n) = n^2(\frac{2}{3} - \frac{3}{n} + \frac{1}{n^2})$ et on remarque que $\lim_{n->\infty}(\frac{2}{3} - \frac{3}{n} + \frac{1}{n^2}) = \frac{2}{3}$ et donc qu'il existe un entier n_0 à partir duquel $\frac{1}{3} < \frac{2}{3} - \frac{3}{n} + \frac{1}{n^2} < 1$, et donc à partir duquel $\frac{n^2}{3} < f(n) < n^2$, ce qui signfie que $f(n) = \theta(n^2)$.

Le même raisonnement appliqué à $g(n) = n\log(n)\frac{1-\frac{1}{n\log(n)}+\frac{1}{n^2}}{1-\frac{1}{n}}$ permet de montrer que $f(n) = \theta(n\log(n))$.

. . . . . .

3 Problème algorithmique (8 points)

On appelle **tableau de classement trié** tout tableau d'entiers positifs $\{T[i]\}_i = 1^n$ tel que
- $T[1] = 1$
- Pour tout $i < n$, $T[i] \leq T[i+1]$, autrement dit T est un tableau trié.
- Pour tout $i \leq n$, $T[i] \leq i$

Un tableau sera dit **tableau de classement** lorsque $Tri(T)$ est un tableau de classement trié, la procédure Tri étant une procédure de tri quelconque. Par exemple, le tableau $(1, 2, 2, 2, 5, 6, 6, 8)$ est un tableau de classement trié, et $(6, 8, 2, 1, 2, 6, 2, 5)$ est un tableau de classement.

1. Ecrire un algorithme $TesteClassementTrie(T, n)$ qui teste si un tableau est un tableau de classement trié. Calculer sa complexité.

TesteClassementTrie

```
if(T[1]<>1) then return(false)
for (int i = 2; i < n+1; i++) {
   if(T[i]<T[i-1] or  T[i]>i) then return(false)}
  }
return(true)
```

L'algorithme est linéaire, puisqu'on exécute une ou deux comparaisons dans chaque corps de boucle.

. . . . . .

2. En déduire la complexité de l'algorithme $TesteClassement(T, n)$ qui consiste tout d'abord à trier le tableau T puis lui appliquer la procédure $TesteClassementTrie$

La procédure de tri est en $n\log(n)$ si on utilise par exemple un quicksort. On réalise ensuite un traitement linéaire, il en résulte don un algorithme en $n + n\log(n)$ soit en $n\log(n)$

. . . . . .

3. Ecrire un algorithme $TesteClassement2(T, n)$ qui teste si un tableau est un tableau de classement sans exécuter de tri et sans utiliser de tableau auxiliaire. Quelle est sa complexité ?

On remarque qu'un tableau est un tableau de classement lorsque pour tout i, le nombre d'éléments du tableau qui sont inférieurs ou égaux à à i ne peut pas être lui strictement supérieur à i.

On commence donc à écrire une fonction NbInférieurs(T,n,x] qui retourne le nombre d'éléments de T qui sont inférieurs ou égaux à x

nb d'éléments inférieurs ou égaux à x

```
nb=0
for (int i = 1; i < n+1; i++) {
   if(T[i]<1+x) then nb++}
  }
return(nb)
```

L'algorithme est linéaire, puisqu'on exécute une comparaison et une opération dans chaque corps de boucle. On en déduit alors la procédure de test suivante.

TesteClassement

```
for (int i = 1; i < n+1; i++) {
   if(NbInférieurs(T,n,i)>i then return(false)}
  }
return(true)
```

L'algorithme est alors quadratique

. . . . . .

4. Ecrire un algorithme $TesteClassementLineaire(T, n)$ qui teste si un tableau est un tableau de classement, de complexité linéaire, en utilisant un tableau auxiliaire.

En utilisant un tableau auxiliaire, on peut pour chaque valeur x présente dans le tableau, incrémenter d'une unité le contenu de la x-ième valeur du tableau auxiliaire. On obtient alors, en une seule passe, c'est à dire en temps linéaire un tableau auxiliaire dans lequel aux[i] est le nombre de valeurs égales à i dans T. On peut, toujours en temps linéaire réaliser les sommes partielles des valeurs de aux, c'est à dire remplacer aux[i] par aux[1]+aux[2]+...+aux[i]. Il suffit enfin, comme dans la question précédente, de vérifier aucune valeur aux[i] n'est supérieure à i. L'algorithme est linéaire, puisqu'on exécute une ou deux comparaisons dans chaque corps de boucle.

. . . . . .

4 Ensembles et Fonctions (6 points)

Soit f la fonction de $\{1, 2, \ldots, 10\}$ qui à un entier n lui fait correspondre l'ensemble de ses diviseurs pairs.

1. Préciser les ensembles de départ et d'arrivée de la fonction f.

L'ensemble de départ est $X = \{1, 2, \ldots, 10\}$ et l'ensemble d'arrivée $\mathcal{P}(X)$

. . . . . .

2. Décrire $f(i)$ pour $i = 1 \ldots 10$. Décrire en extension (par ses éléments) l'ensemble $A = \{x, x \notin f(x)\}$

On a $f(1) = \emptyset$, $f(2) = \{2\}$, $f(3) = \emptyset$, $f(4) = \{2, 4\}$, $f(5) = \emptyset$,$f(6) = \{2, 6\}$, $f(7) = \emptyset$, $f(8) = \{2, 4, 8\}$, $f(9) = \emptyset$, $f(10) = \{2, 10\}$. On en déduit que $A = \{1, 3, 5, 7, 9\}$

. . . . . .

3. Montrer qu'il ne peut pas exister d'entier a tel que $f(a) = A$.

Alors A ne contient que des nombres impairs et ne peut pas être égal à un $f(a)$ qui ne contient que des nombres pairs. Plus généralement, si il existait un entier a tel que $A = f(a)$, alors par définition de A si a était dans A alors il ne serait pas dans $f(a) = A$ d'où une contradiction. Et on obtient une contradiction analogue si a n'était pas dans A : alors a ne serait pas dans $f(a) = A$.

. . . . . .

5 Dénombrabilité (4 points)

1. L'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ est il dénombrable ? Justifier votre réponse.

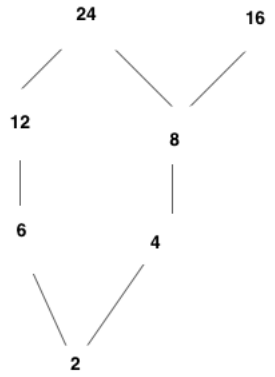

FIGURE 1 – Diagramme de Hasse

L'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ contient l'ensemble des applications de $\mathbb{N}$ dans $\{0,1\}$, c'est à dire des suites infinies de 0 et de 1. Or, à toute suite infinie de 0 et de 1 on peut lui faire correspondre la partie de $\mathbb{N}$ constituée des entiers qui ont pour image 1. C'est donc que l'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ est plus grand que $\mathcal{P}(\mathbb{N})$, qui est non dénombrable et donc L'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ est non dénombrable.

Une autre preuve utilise le procédé de diagonalisation de Cantor. On suppose que l'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ est dénombrable et qu'il s'écrit donc $\{f_k\}_{k\in\mathbb{N}}$. On construit alors l'application g, de $\mathbb{N}$ dans $\mathbb{N}$ définie par $g(k) = f_k(k) + 1$. Par définition, cette fonction g ne peut pas être l'une des fonction f_k ce qui est contradictoire avec l'hypothèse.

. . . . . .

2. L'ensemble des applications de $\{0,1\}$ dans $\mathbb{N}$ est il dénombrable ? Justifier votre réponse.

Une application de $\{0,1\}$ dans $\mathbb{N}$ est définie par les images de 0 et de 1, c'est à dire par deux entiers. L'ensemble des applications de $\{0,1\}$ dans $\mathbb{N}$ est donc en bijection avec $\mathbb{N}\times\mathbb{N}$ qui est dénombrable.

. . . . . .

6 Relations (4 points)

1. Dessiner le diagramme de Hasse de la relation divise sur l'ensemble $A = \{2,4,6,8,12,16,24\}$.

Voir Figure 1.

. . . . . .

2. Préciser si l'ensemble A admet un plus grand élément, un plus petit élément, une borne supérieure, une borne inférieure.

L'ensemble A admet un plus petit élément 2 qui est aussi borne inférieure. Il a deux éléments maximaux qui sont 24 et 16, et une borne supérieure qui est 48, le ppcm de 24 et 16.

. . . . . .