400 099 IF - 2012
S1 - 1S

BU120194

Université Lille1

UFR IEEA

Examen AAC 1ère session
Documents autorisés

M1 Informatique

Bonne Année 2012!

Exercice 1 : Compétences de base

Remarque: pour cet exercice, des points seront retirés si le nombre de réponses incongrues est déraisonnable.

Q 1. Trier les fonctions $n^4+\sqrt{n}, 5*n+\log n, \log n, n^3+2n^2, 4n^2+4n, n\log n, 3^n, \log n+\sqrt{n}, 4^n$ suivant leur ordre de grandeur asymptotique: f sera "avant" g si $f \in O(g)$. Par exemple, $n, 1, n^2, \log n$ serait trié en $1, \log n, n, n^2$.

Q 2. Répondre en justifiant très brièvement aux questions suivantes:

Q 2.1. Toute propriété P est-elle décidable?

Q 2.2. Toute propriété NP est-elle décidable?

Q 2.3. Toute propriété P est-elle NP?

Q 2.4. Toute propriété NP est-elle P?

Q 2.5. Si R est une propriété NP, "*non* R" est-elle une propriété NP?

Q 2.6. Toute propriété NP peut-elle être décidée par un algorithme exponentiel?

Q 2.7. Votre collègue prétend avoir trouver un algorithme polynomial pour décider une propriété NP-dure. Qu'en pensez-vous?

Q 2.8. Votre collègue prétend avoir trouver un algorithme polynomial pour décider une propriété NP. Qu'en pensez-vous?

Q 2.9. Toute propriété P peut-elle être réduite polynomialement en une propriété NP-complète?

Q 2.10. Si L_1 et L_2 sont deux langages récursifs, le concaténé des deux langages $L_1.L_2$ est-il un langage récursif?

Q 3. Soit un problème d'optimisation où il s'agit de minimiser une fonction objectif, et deux heuristiques $H1$ et $H2$ (qu'on suppose correctes, c.à.d. donnant toutes les deux une solution non nécessairement optimale mais correcte!). Le ratio de garantie de $H1$ est 2, celui de $H2$ est 5.

Q 3.1. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 11 alors qu'$H2$ donne une solution de coût 2? Justifier.

Q 3.2. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 6? Justifier.

Q 3.3. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 45? Justifier.

Q 3.4. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 60? Justifier.

Exercice 2 : Le nombre d'occurrences

Soit un tableau T de n entiers triés en ordre croissant et a un entier: on recherche dans le tableau T le nombre d'occurrences de a.

Donnée: n entier, T un tableau de n entiers triés dans l'ordre croissant, a un entier.

Sortie : Le nombre de i tel que $a = T[i]$

Par exemple, si $n = 8$ et $T[0] = 1, T[1] = 6, T[2] = 12, T[3] = 12, T[4] = 12, T[5] = 24, T[6] = 26, T[7] = 26$, alors pour $a = 12$, on retournera 3, pour $a = 20$ on retournera 0, pour $a = 26$, on retournera 2.

Q 1. Proposer un algorithme en $O(log\ n)$ pour le problème. Justifier qu'il est correct et que sa complexité est bien en $O(log\ n)$.

Exercice 3 : Monnaie, le retour

Soit la version suivante du problème du monnayeur, où on peut utiliser au plus une seule fois chacune des pièces:

Donnée:
un entier positif n, le nombre de types de pièces
$v_0, v_1, \ldots, v_{n-1}$, n entiers positifs, les valeurs faciales
S une somme à payer.

Sortie : Oui si on peut payer S, en utilisant **au plus** un seule fois chacune des pièces, Non, sinon.

Par exemple,pour $n = 4$, $v_0 = 1, v_1 = 2, v_2 = 8, v3 = 10$, la réponse sera oui pour $S = 19(= 1+8+10)$, ou $S = 9 = (1+8)$ mais non pour $S = 16$.

Q 1. Proposer un algorithme en $O(n*S)$ pour le problème.

Q 2. Modifier l'algorithme précédent pour qu'il retourne également la façon de payer S.

Exercice 4 : Couverture

Soit X un ensemble fini de n intervalles de la forme $[d_i, f_i]$, d_i et f_i étant réels. On dira qu'un ensemble de réels P couvre X, si pour tout intervalle de X, il existe un réel de P dans l'intervalle. On cherche un ensemble (fini) de cardinal minimal qui couvre X.

Par exemple, si $X = \{[2,12],[7,10],[4,8],[6,16],[14,18],[34,40],[16,30]\}$, on pourrait prendre $P = \{7,17,35\}$ ou $P = \{7.8,18,39.2\}$

Le problème est donc:

Donnée:
un entier n
D et F, deux tableaux de n réels représentant les d_i et f_i.
Sortie:
P un ensemble fini de réels de cardinal minimal tel que: pour tout i $(0 \leq i < n)$, $P \cap [d_i, f_i] \neq \emptyset$

Q 1. Proposer un algorithme polynomial pour le problème. Evaluer la complexité de l'algorithme et justifier que l'algorithme est correct.

Exercice 5 : Partage

Le problème de décision 3−Partition consiste en répartir en trois ensembles des entiers donnés de façon à ce que les sommes des trois ensembles soient identiques. Formellement, $3-Partition$ est défini par:

Donnée:
n -un entier
x - un tableau de n entiers
Sortie:
Oui Ssi il existe J, K, L trois sous-ensembles de $[1..n]$ tels que

1. $J \cap K = J \cap L = K \cap L = \emptyset$ et $J \cup K \cup L = [1..n]$, i.e. $\{J, K, L\}$ partition de $[1..n]$
2. $\sum_{i\in J} x[i] = \sum_{i\in K} x[i] = \sum_{i\in L} x[i]$

Q 1. Combien de partitions de $[1..n]$ en 3 sous-ensembles existe-t-il?

Q 2. Montrer que 3−Partition est NP.

Q 3. On rappelle que 2−Partition est NP-dur, avec 2−Partition défini par:

Donnée:
n -un entier
x - un tableau de n entiers
Sortie:
Oui Ssi il existe J, K deux sous-ensembles de $[1..n]$ tels que

BU120194

1. $J \cap K = \emptyset$ et $J \cup K = [1..n]$
2. $\sum_{i \in J} x[i] = \sum_{i \in K} x[i]$

Q 3.1. Montrer que 2−Partition se réduit polynomialement en 3−Partition.
Q 3.2. Qu'en déduire pour 3−Partition?
Q 3.3. Pensez-vous que 3−Partition se réduit polynomialement en 2−Partition? Justifier brièvement.
Q 4. On s'intéresse maintenant au problème d'optimisation. On suppose ici que tous les entiers $x[i]$ sont positifs ou nuls. On cherche donc à partitionner les $x[i]$ en trois, de façon la plus équilibrée possible.
Le problème $3 - PartitionOpt$ est donc défini par:

Donnée:
n −un nb d'entiers
x − un tableau de n entiers
Sortie:
J, K, L trois sous-ensembles de $[1..n]$ tels que

1. $J \cap K = J \cap L = K \cap L = \emptyset$
2. $J \cup K \cup L = [1..n]$
3. $max(\sum_{i \in J} x[i], \sum_{i \in K} x[i], \sum_{i \in L} x[i])$soit minimal.

Q 4.1. D'après ce qui précède, que pensez-vous de la complexité du problème $3 - PartitionOpt$?
Q 4.2. Justifier qu'il existe un algorithme exponentiel pour $3 - PartitionOpt$.
Q 5. On suppose ici qu'on dispose d'une classe Ensemble pour implémenter les ensembles d'entiers, avec $ajout(x)$ qui permet d'ajouter un entier, et $somme()$ qui calcule la somme des éléments de l'ensemble.
Soit l'algorithme:

```
// x tableau de n entiers
Ensemble I,J,K;
//I,J,K ensembles initialises à l'ensemble vide
void Trois_Part(){
Trier  x par valeurs décroissantes;
for (i=0;i<x.length;i++){
    if (I.somme() <= J.somme())
           if  (I.somme() <=K.somme()) I.ajout(x[i]);
           else K.ajout(x[i]);
    else
           if  (J.somme() <=K.somme()) J.ajout(x[i]);
           else K.ajout(x[i];}
```

Q 5.1. Montrer par un exemple que l'algorithme ne donne pas toujours la solution optimale.
Q 5.2. Montrer que si n est inférieur ou égal à 4, l'heuristique est optimale.
Q 5.3. Montrer que l'algorithme offre une garantie de 8/5.
Remarque: la preuve -correcte- d'une garantie différente de 8/5 sera prise en compte.