Université de Lille 1 SCIENCES ET TECHNOLOGIES
UFR IEEA

ACT
Durée 3h
Tous documents autorisés

2014–2015
Master1 Informatique
9 janvier 2015

Meilleurs voeux pour l'année 2015.

Les algorithmes seront écrits en pseudo-langage ou dans un langage de votre choix (C, Java, Caml). La clarté de vos réponses et de votre code sera prise en compte. Le barème indiqué (sur 22) est le barème minimal.

Exercice 1 : Quelques questions d'application immédiate du cours - *6 pts* -

Q 1. *À ordonner - 1 pt* - On veut trier des fonctions par "ordre" asymptotique de grandeur: si $f = o(g)$, on dira que $f < g$. Par exemple $\sqrt{n} < n^5 < n^8$. Trier les fonctions suivantes:

$$n^3 \quad 3^n \quad \lg n \quad n^2 \quad n\sqrt{n} \quad n^n$$

Q 2. *Vrai ou Faux - 1 pt* - Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse - il n'est pas demandé de justifier, mais les réponses fausses pourront être comptées "négativement":

1. $n^3 + n^4 \in O(n^5)$?
2. $n^4 \in O(n^3)$?
3. $n^2 \log n \in O(n^3)$?
4. $n^2 + \log n \in O(n^2)$?
5. $n^2 + \log n \in \Theta(n^2)$?
6. $n^2 \log n \in O(n^2)$?

Q 3. *Analyse de complexité - 1 pt* - Pour chacun des deux algorithmes suivants, dire si sa complexité (temporelle) dans le pire des cas est en $\Theta(log\ n)$, $\Theta(n)$, $\Theta(n\ log\ n)$ ou $\Theta(n^2)$, en justifiant brièvement:

```
int T1(int n){
  int c=0;
  for (int i=1;i<n;i=i*10)
    for (int j=n;j>0;j=j-10) c++;
  return c;}
```

```
int T2(int n){
  int c=0;
  for (int i=10;i>0;i--)
    for (int j=1; j<n*i;j++) c++;
  return c;}
```

Q 4. *Justification - 0.5 pt* - Vous n'avez pas réussi à obtenir un algorithme polynomial pour le problème (qu'on supposera de type OUI/NON) que vous aviez à résoudre. Pour justifier votre échec, que pouvez-vous essayer de prouver?

Q 5. *Réductions polynomiales - 1.5 pts* - La professeur Therorica a demandé à ses étudiants d'étudier la complexité de six problèmes de décision P_1, P_2, P_3, P_4, P_5, P_6. Le partage du travail étant une réalité quotidienne pour ces étudiants, ils décident de rassembler le fruit de leurs efforts. Chacun énonce donc ce qu'il a réussi à prouver:

Etudiant 1: Le problème P_1 est NP-dur.
Etudiant 2: Le problème P_2 se réduit polynomialement dans le problème P_1.
Etudiant 3: Le problème P_1 se réduit polynomialement dans le problème P_3.
Etudiant 4: Le problème P_3 se réduit polynomialement dans le problème P_4.
Etudiant 5: Le problème P_5 se réduit polynomialement dans le problème P_2.
Etudiant 6: Le problème P_6 se réduit polynomialement dans le problème P_5.
Etudiant 7: Le problème P_5 est P.

Bien sûr, leurs preuves ne sont pas à mettre en doute.

Q 5.1. - *1 pt* - Parmi $P_1, P_2, P_3, P_4, P_5, P_6$, quels sont les problèmes dont on peut affirmer qu'ils sont NP-durs? P? Justifier brièvement.

Q 5.2. - *0.5 pt* - Un étudiant arrive et affirme avoir prouvé que P_3 est P. Qu'en pensez-vous?

Q 6. *Récursif? - 1 pts* - Pour chaque affirmation suivante, dire si elle est vraie ou fausse. Justifiez brièvement.

1. L'ensemble des mots sur $\{a, b\}$ ayant autant de a que de b est récursif.
2. Si L_1 ou L_2 est récursif, $L_1 \cup L_2$ est récursif.
3. Si $L_1 \cup L_2$, $L_1 \cap L_2$ et L_1 sont récursifs, alors L_2 est récursif.

Exercice 2 : Carré Noir - *5 pts* -

Soit une matrice A de taille n × n où les coefficients valent 0 ou 1 -qui peut par exemple représenter une image en noir et blanc-. On cherche un carré de largeur maximum qui ne contient que des 1.

Par exemple, soit la matrice A :

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

.

Alors la largeur maximum d'un carré qui ne contient que des 1 est 4 et il y a un seul tel carré, son coin en haut à gauche étant à la position $(1, 2)$:

	0	1	2	3	4	5	6
0	1	0	0	1	1	1	0
1	0	0	**1**	**1**	**1**	**1**	0
2	1	0	**1**	**1**	**1**	**1**	1
3	0	1	**1**	**1**	**1**	**1**	0
4	1	0	**1**	**1**	**1**	**1**	0
5	1	1	1	1	1	1	0
6	0	1	1	0	1	1	0

Le problème est donc:

Entrée : une matrice A de taille $n \times n$ où les coefficients valent 0 ou 1.

Sortie : la largeur maximum k d'un carré de 1 dans A, ainsi que les coordonnées (l, c) du coin en haut à gauche d'un tel carré.

Pour l'exemple précédent la sortie sera donc 4 et $(1, 2)$.

Q 1.- *1 pt* - Quelle serait la complexité d'un algorithme naïf qui testerait tous les carrés?

Q 2.- *4 pts* - L'objectif est de proposer un algorithme de type "programmation dynamique" en $O(n^2)$ pour résoudre le problème.

Appelons $larg(l, c)$ la largeur d"un carré maximal de 1 à la position (l, c).

Q 2.1. *Cas simple*- Si $A(l, c) = 0$, que vaut $larg(l, c)$?

Q 2.2. *Cas de base*- Quelle est la largeur d'un carré maximal de 1 à une position de type $(n-1, c)$ ou $(l, n-1)$?

Q 2.3. *Décomposition du problème*- Soit $0 \leq l < n-1$ et $0 \leq c < n-1$.

Supposons qu'à la position (l, c) il y ait un carré de 1 de largeur k, avec $k \geq 2$. Par quelle valeur peut-on alors minorer $larg(l+1, c)$? $larg(l, c+1)$? $larg(l+1, c+1)$?

Si $A(l, c) = 1$, exprimer $larg(l, c)$ en fonction de $larg(l+1, c), larg(l, c+1)$ et $larg(l+1, c+1)$.

Q 2.4. *Algorithme*- Déduire de ce qui précède un algorithme de programmation dynamique en $O(n^2)$ pour résoudre le problème.

Exercice 3 : Pochettes surprises - *6 pts* -

On dispose de n gadgets et on connait leurs valeurs respectives. On souhaite faire le plus possible de pochettes surprises avec ces gadgets avec la contrainte que la valeur des objets contenus dans chaque pochette soit au moins M. Par exemple, si $n = 10$, et si les valeurs sont $3, 5, 4, 3, 2, 5, 4, 5, 2, 5$, pour $M = 6$, on peut faire cinq pochettes comme suit: {3, 5}, {4, 2}, {4,5}, {5,2}, {3,5} (On représente ici un gadget par sa valeur).

Le problème *Poc* sera donc:

Entrée:

n – entier >0, nombre d'objets

$v_0, \ldots, v_{n-1}$ –entiers >0, les valeurs des objets

M, entier >0, la valeur minimale que doit avoir une pochette

Sortie: une répartition des n objets en pochettes surprises telle que chaque pochette soit de valeur au moins M et qui maximise le nombre des pochettes.

Formellement une répartition peut être définie par:
$aff : [0..n-1] \rightarrow [1..n]$: au plus n pochettes, si on a n objets
telle que:
$aff([0..n-1]) = [1..k]$ pour un certain k : les pochettes sont numérotées de 1 à k et si on a une pochette de numéro $i > 1$, on a une pochette de numéro $i-1$. k est donc le nombre de pochettes.

On cherche donc une répartition telle que, pour tout p entre 1 et k, $\sum_{i/aff[i]=p} v_i \geq M$ - chaque pochette a une valeur au moins M, et telle que k soit maximal - la fonction objectif à maximiser est le nombre de pochettes.

Q 1.- *0.5 pt* - Dans l'exemple précédent peut-on faire plus de 5 pochettes? Justifier.

Soit le problème *PocDec* de décision associé à *Poc*:
Entrée:
n – entier >0, nombre d'objets
$v_0, \ldots, v_{n-1}$ –entiers >0, les valeurs des objets
M, entier >0, la valeur minimale que doit avoir une pochette
k, entier >0, le nombre de pochettes "cible"
Sortie: Oui si il existe une répartition des objets en k pochettes surprises telle que chaque pochette soit de valeur au moins M, Non sinon.
Q 2.- *1 pt* - Montrer que *PocDec* est NP.
Q 3.- *2 pts* - Montrer que *PocDec* est NP-dur. Vous pourrez utiliser les propriétés prouvées NP-dures en cours ou TP.
Q 4.- *0.5 pt* - Justifier que le problème initial *Poc* est NP-dur.
Q 5.- *2 pts* - Proposer une heuristique de garantie (au plus) deux pour le problème *Poc*. Justifier.
Aide: Vous pourrez dans un premier temps supposer que tous les gadgets ont une valeur inférieure à M.

<u>Exercice 4</u> : Histoire de Cliques - *5 pts* -

Rappel: dans un graphe, une *clique* est un ensemble de sommets qui sont **tous** reliés deux à deux- chaque paire est reliée. Le problème Clique consiste à déterminer s'il existe dans un graphe donné une clique de cardinal égal à un entier k donné. Le problème de décision Clique est donc défini par:

Entrée:
$G = (S, A)$ - -un graphe
k - -un entier
Sortie: Oui, si il existe une clique de cardinal k, i.e. une partie C de S de cardinal k telle que toute paire de sommets de S soit reliée par un arc de A, Non sinon.

Clique est connu NP-complet.

Soient les trois problèmes suivants:

1000-Clique
Entrée: $G = (S, A)$ –un graphe
Sortie: Oui, si il existe une clique de cardinal égal à 1000, Non sinon.

Double-Clique
Entrée:
$G = (S, A)$ –un graphe
k - -un entier
Sortie: Oui, si il existe deux cliques distinctes de cardinal égal à k, Non sinon.

Presque-Clique
Entrée: $G = (S, A)$ –un graphe.
Sortie: Oui, si il existe une clique de cardinal égal à $|S| - 1$ ($|S|$ est le cardinal de S), Non sinon.
Q 1. - *4 pts* - Parmi ces trois problèmes, 1000-Clique, Double-Clique, Presque-Clique, quel est le problème NP-dur, quels sont les problèmes P? Justifier.
Q 2. - *1 pt* - Les trois problèmes sont-ils NP? Justifier.