Université Lille1 Sciences et Technologies
UFR IEEA

Examen AAC 1ère session
Documents autorisés

14 janvier
M1 Informatique

Remarque: le sujet couvre tout le programme et est par conséquent un peu long: le barème en tiendra compte.

Exercice 1 : Compétences de base

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

Q 1. Trier les fonctions $5^n, n^3, n^2+n, n+5, n\log n, 2^n, \sqrt{n}$ 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. Pour chacun des algorithmes suivants, dire si sa complexité (temporelle) dans le pire des cas est en $\Theta(\log n)$, $\Theta(n)$ ou $\Theta(n^2)$, en justifiant très brièvement:

```
int T1(int n){
  if (n <= 0) return 1;
  else return (1+ T1 (n-4));}

int T2(int n){
  int c=0;
  for (int i=1;i<4*n;i=4*i) c++;
  return c;}
```

```
int T3(int n){
  if (n <= 0) return 1;
  else return (1+ T3 (n/4));}

int T4(int n){
  if (n <= 0) return 1;
  else return (T4(n/2)+T4(n/2)+T4(n/2)+T4(n/2));}
```

Q 3. *La classe NP*

Q 3.1. La propriété de décision *Dec* a été prouvée NP-complète. Que pensez-vous de chacune des trois affirmations suivantes? Justifiez brièvement.

1. Il existe un algorithme non déterministe polynômial pour décider *Dec*.
2. Il n'existe pas d'algorithme déterministe polynômial pour décider *Dec*.
3. La propriété *Dec* est décidable.
4. Il existe un algorithme déterministe pour décider *Dec*.

Q 3.2. Soient P_1, P_2, P_3, Q des problèmes de décision.
Supposons qu'on sache que P_1 est P, P_2 est NP-dur, P_3 NP.
Pour chacune des affirmations suivantes, dire si elle est "Vraie", "Fausse" (pas de justification demandée):

(Af1) P_3 se réduit en P_2

(Af2) P_1 se réduit en P_2

(Af3) Si Q se réduit polynômialement en P_2, Q est $NP-$dur.

(Af4) Si P_3 se réduit polynômialement en Q, Q est $NP-$dur.

(Af5) Si P_2 se réduit polynômialement en Q, Q est $NP-$dur.

(Af6) Si Q se réduit polynômialement en P_3, Q se réduit polynômialement en P_2.

Q 4. 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 3, celui de $H2$ est 6.
Q 4.1. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 2? Justifier.
Q 4.2. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 5? Justifier.
Q 4.3. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 15? Justifier.
Q 4.4. Peut-on avoir, pour une certaine donnée, $H1$ qui donne une solution de coût 10 alors qu'$H2$ donne une solution de coût 80? Justifier.
Q 5. Pour chaque affirmation suivante, dire si elle est vraie ou fausse. Justifier brièvement.

1. Tout langage reconnu par un automate fini peut être décidé par une Machine de Turing.
2. L'ensemble des mots ayant deux fois plus de a que de b est récursif.
3. L'intersection de deux langages algébriques n'est pas un langage récursif.
4. Le complémentaire d'un langage algébrique est récursif.
5. Le complémentaire d'un langage algébrique est récursivement énumérable.
6. Si L_1 n'est pas récursif et L_2 reconnaissable, $L_1 \cup L_2$ n'est pas récursif.

<u>Exercice 2</u> : Encore une histoire de découpage

Le but est de diviser l'intervalle $[0, m]$ en morceaux dont la longueur est la plus proche possible d'une longueur l_0. On impose de plus la contrainte que les découpes soient effectuées à des endroits "prédécoupés" $0 < x_1 \ldots < x_n < m$. Le "coût" du découpage est ici $\sum_{i=1}^{k}(y_i - l_0)^2$, si il y a k morceaux et si les longueurs des morceaux sont $y_1, ...y_k$.

On doit donc choisir un sous-ensemble de points parmi les n points x_i, $x_{i_1} < x_{i_2} < \ldots < x_{i_{k-1}}$, qui minimise le coût du découpage associé, soit $\sum_{i=1}^{k}(y_i - l_0)^2$ avec $y_i = x_i - x_{i-1}$ (on pose $x_{i_0} = 0, x_{i_k} = m$). On remarquera que le nombre de morceaux n'est a priori pas fixé.

Exemple: le découpage optimal de $[0, 650]$ avec $l_0 = 200$ et $x_1 = 100, x_2 = 350, x3 = 400, x_5 = 500$ est x_1, x_2, x_5.
Q 1. Quel sera le découpage optimal de $[0, 650]$ avec $l_0 = 200$ et $x_1 = 100, x_2 = 250, x3 = 450, x_5 = 600$?

Soit $C(i)$ le coût d'un découpage optimal du segment $[x_i, m]$ ($C(0)$ est le coût d'un découpage optimal du segment $[0, m]$); On cherche donc $C(0)$ et le découpage correspondant.
Q 2. Que vaut $C(n)$?
Q 3. Pour $0 \leq i < n$, définir la relation de récurrence qui définit $C(i)$ en fonction des $C(j)$, $j > i$.
Q 4. En déduire un algorithme récursif naïf qui calcule $C(0)$. Evaluer l'ordre de grandeur de sa complexité.
Q 5. *Les deux questions suivantes peuvent être résolues simultanément:*
Q 5.1. Transformer l'algorithme précédent en utilisant le principe de la programmation dynamique. Quelle est la complexité de ce nouvel algorithme?
Q 5.2. Modifier l'algorithme pour calculer aussi le découpage (et pas seulement son coût).

<u>Exercice 3</u> :

Soient un ensemble d'intervalles $G = \{I_1, \ldots, I_n\}$ et un ensemble d'entiers $R = \{e_1, e_2, \ldots, e_m\}$. Chaque intervalle I est spécifié par deux entiers, ses bornes gauche et droite, $g(I), d(I) : I = [g(I), d(I)]$. On dit qu'un intervalle couvre un entier si l'entier appartient à l'intervalle;

Le but est de choisir un nombre minimal d'intervalles parmi G qui couvrent tous les points de R, si cela est possible.

Exemple: Soient $G = \{[30,58],[65,80],[40,75],[20,55]\}$ et $R = \{25,35,55,65,75,79\}$. Alors, une façon optimale de couvrir R à partir de G est $\{[65,80],[20,55]\}$. Par contre, si $G = \{[30,58],[65,80],[20,55]\}$ et $R = \{25,35,55,64\}$, il n'y a pas de solution car 64 n'est pas couvert;

Les intervalles sont a priori tous distincts (G est un ensemble) et de plus, on supposera qu'aucun intervalle n'en contient un autre, c.à.d:

. si $g(I) = g(I')$, alors $I = I'$

. si $g[I] < g[I']$, alors $d[I] < d[I']$.

Par exemple, on ne peut avoir dans G à la fois $[3,6]$ et $[4,5]$. On peut remarquer que cela implique que tous les intervalles ont des bornes gauches (resp. droites) différentes.

Le problème est donc:

Entrée:

$G = \{I_1, \ldots, I_n\}$, un ensemble d'intervalles (tels qu'aucun n'en contienne un autre)

$R = \{e_1, e_2, \ldots, e_m\}$, un ensemble d'entiers.

Sortie:

. $Couv$, un sous-ensemble de G de cardinal minimal tel que tout élément de R soit couvert par un intervalle de $Couv$, si c'est possible.

. "Message d'**Erreur**" si ce n'est pas possible.

G et R ne sont pas triés mais vous pouvez supposer que G et R sont représentés par la structure de données qui vous convient et que vous disposez des méthodes "élémentaires" sur les intervalles qui vous conviennent: intersection de deux intervalles, inclusion, accès aux bornes, comparaison selon une des deux bornes, appartenance d'un réel à un intervalle... Vous pourrez supposer que leur coût est en $O(1)$, si cela est raisonnable!

Q 1. Soient $G = \{[30,58],[65,80],[40,75],[20,55]\}$ et $R = \{25,35,55,60,75\}$. Proposer une façon optimale de couvrir R à partir de G.

Q 2. Proposer un algorithme glouton exact en $O(n \log n + m \log m)$ pour le problème. Justifier.

Q 3. On suppose toujours que les intervalles sont tous distincts mais on ne suppose maintenant plus qu'aucun intervalle ne contient l'autre, c.à.d on peut avoir $g[I] \leq g[I']$ et $d[I] \geq d[I']$. Par exemple, on peut avoir simultanément $[3,6]$, $[3,7]$ et $[4,5]$.

Proposer un prétraitement de G en $O(n \log n)$ pour extraire de G tous les intervalles maximaux, i.e. qui ne sont contenus dans aucun autre intervalle.

Exemple: De $G = \{[30,58],[35,50],[40,75],[20,55],[26,55]\}$, on extraira $G = \{[30,58],[40,75],[20,55]\}$.

Exercice 4 : Coloriage

Soit $3-COL$ le problème du 3-coloriage de graphes défini par:

Entrée : $G = (S, A)$ un graphe non-orienté

Sortie :

Oui, si il existe un coloriage correct de G en 3 couleurs, i.e. une application $coul : S \to [1..3]$ telle que $\forall x, y \in S,\ (x,y) \in A \Longrightarrow coul(x) \neq coul(y)$.

Non, sinon.

Soit $4-COL$ le problème du 4-coloriage de graphes défini par:

Entrée : $G = (S, A)$ un graphe non-orienté

Sortie :

Oui, si il existe un coloriage correct de G en 4 couleurs, i.e. une application $coul : S \to [1..4]$ telle que $\forall x, y \in S,\ (x,y) \in A \Longrightarrow coul(x) \neq coul(y)$.

Non, sinon.

Rappel: $3-Col$ est NP-complet.

Q 1. Montrer que le problème $4-COL$ est NP.

Q 2. Proposer une réduction polynômiale de $3-Col$ dans $4-Col$. Justifier.

Q 3. Qu'en déduire pour $4-Col$?

Q 4. Montrer, **sans exhiber la réduction**, que $4-Col$ se réduit polynômialement dans $3-Col$.

Exercice 5 : Indépendance

Soit *IndependantOpt* le problème défini par:

Entrée : $G(S, A)$ un graphe non-orienté

Sortie : Un sous-ensemble indépendant de S de cardinal maximal.

Rappel: Un sous-ensemble de sommets est indépendant si aucun arc ne relie deux sommets de l'ensemble et le problème de décision *Independant* défini ci-dessous est $NP-$complet:

Entrée : $G(S, A)$ un graphe non-orienté, k un entier

Sortie : Oui, si il existe un sous-ensemble indépendant de S de cardinal k, Non sinon

Q 1. Que pensez-vous de la complexité du problème *IndependantOpt*? Justifier.

Q 2. Soit l'heuristique suivante pour *IndependantOpt*:

```
Initialiser Ind à l'ensemble vide;
tant que S est non vide faire
  choisir s dans S;
  l'ajouter à Ind;
  enlever de S, s et tous ses voisins;
fin tant que;
retourner Ind;
```

Q 2.1. Montrer que cette heuristique n'offre pas de garantie.

Q 2.2. *Rappel:* le degré du sommet d'un graphe est égal à son nombre de voisins. Le degré d'un graphe G, noté $d(G)$, est le maximum des degrés de ses sommets.

Si $d(G)$ est nul, que dire du sous-ensemble indépendant de taille maximale de G? Que retourne l'heuristique?

Q 2.3. Supposons maintenant que $d(G)$ soit non nul. Soit $H(G)$ le sous-ensemble produit par l'heuristique et $Opt(G)$ un sous-ensemble indépendant de taille maximale de G. Justifier que $|Opt(G)| \leq d(G) * |H(G)|$, avec $d(G)$ le degré du graphe, $|A|$ le cardinal de A.

Pourquoi cela ne contredit-il pas le fait que l'heuristique n'offre pas de garantie?

Q 3. Que pensez-vous de la complexité du problème lorsqu'il est restreint aux graphes de degré au plus 2? Justifier.