Université Lille1 Sciences et Technologies

UFR IEEA

Master d'Informatique-S1

10 janvier 2014
Documents autorisés

AAC

Bonne Année 2014

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 est le barème "minimal".

Exercice 1 : Vrai ou Faux -*3 pts*

Pour chacune des affirmations suivantes, dites si elle est vraie ou fausse et justifiez brièvement.

1. L'union de deux langages algébriques est un langage récursif.
2. L'union de deux langages non récursifs est un langage non récursif.
3. Le complémentaire d'un langage non récursif est un langage non récursif.
4. Une propriété NP est décidable.
5. Si Q est une propriété NP, *non* Q est une propriété NP.
6. Toute propriété NP peut être décidée par un algorithme déterministe.

Exercice 2 : Somme maximale -*5 pts*

Soit un tableau T de n entiers. On souhaite extraire de ce tableau des entiers tels que :
— on ne peut choisir deux entiers consécutifs dans le tableau (i.e. T[i] et T[i+1]).
— la somme des entiers choisis soit maximale.

Exemple : supposons que le tableau contient dans cet ordre $4, 3, 7, 12, 2, 5, 9, 1$; pour respecter la première contrainte, on peut choisir par exemple $4, 7, 2, 9$ ou $4, 12, 5, 1$ ou $3, 12, 1$ ou ou $4, 12, 9$. C'est $4, 12, 9$ qui maximise la somme.

Q 1. - *0,5 pt* - Pour $30, 50, 100, 20$, quelle est la solution optimale ? Même question pour $40, 50, 45, 50, 60, 100, 60$.

Q 2. - *0,5 pt* - Soit l'algorithme glouton naïf suivant :

```
/* T tableau de n entiers */
int i=0;
int s=0;
while (i<n) {
if (T[i]>T[i+1])
  {s=s+T[i];System.out.print(T[i]); i=i+2;}
else
  {s=s+T[i+1];System.out.print(T[i+1]); i=i+3;}
}
```

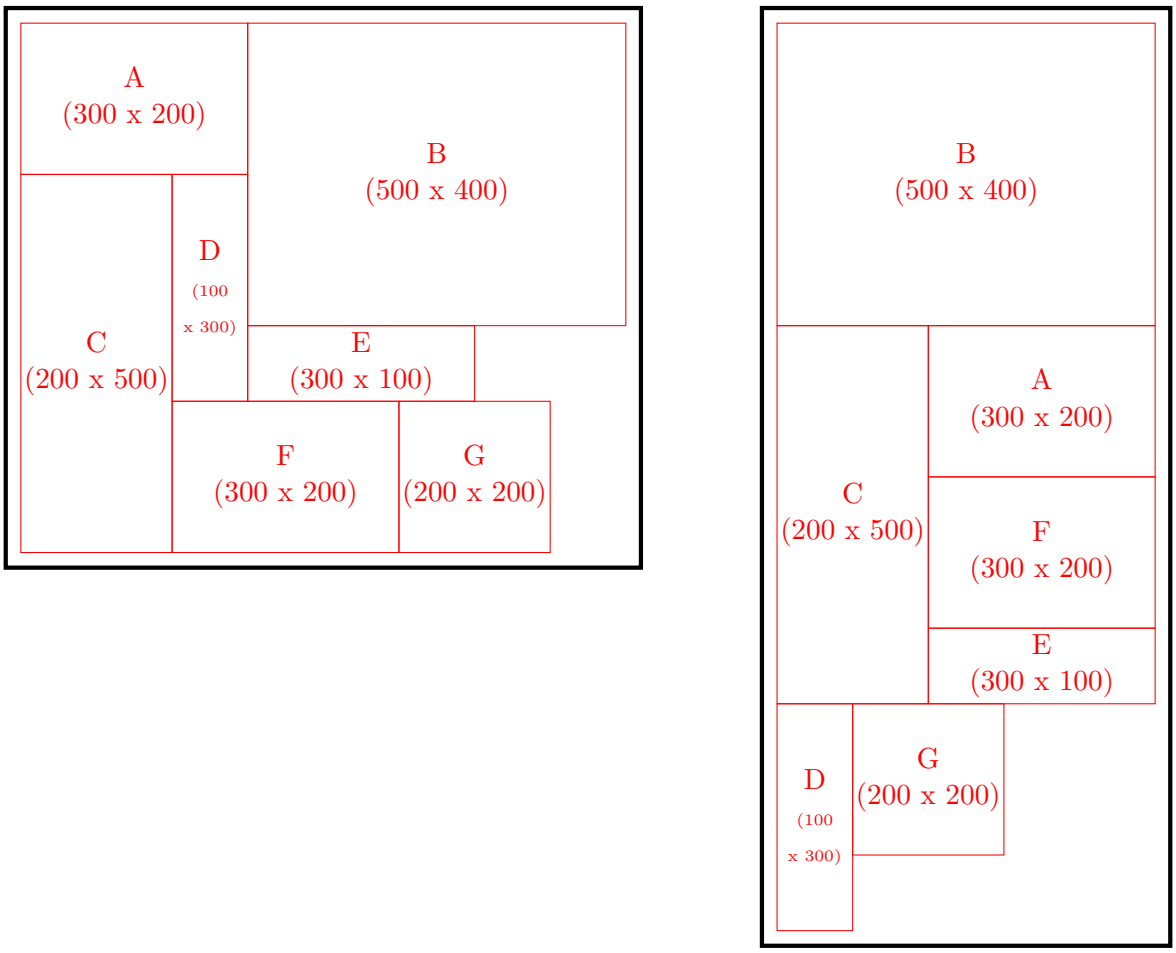
Pensez-vous que cet algorithme retourne toujours la somme maximale ? Justifiez.

Q 3. - *4 pts* - Proposer un algorithme en $\Theta(n)$ qui calcule la somme optimale et sort la (ou une, si il y a plusieurs possibilités) liste des entiers correspondants.

Exercice 3 : Responsive Design -*12 pts*

Sur le Web, la mode est au *Responsive Design*, c'est-à-dire à ce que la présentation d'une page web s'adapte automatiquement à la largeur de l'écran. L'objectif c'est que la même page web, le même code HTML, présente correctement sur un ordinateur (largeur $\approx$ 2000 pixels) sur une tablette (largeur $\approx$ 1000 pixels) et sur un smart-phone (largeur $\approx$ 300 pixels). Pour

cela, le contenu d'une page web est généralement organisé dans des *blocs* rectangulaires dont les dimensions sont fixées. En fonction de la largeur de l'écran, les blocs peuvent être disposés différemment. Dans le dessin ci-dessous, on voit deux exemples de réagencements des blocs d'une page web en fonction de la largeur de l'écran.



Dans la plupart des cas, des méthodes simplistes sont utilisées pour placer les blocs (par exemple, les blocs qui débordent à droite passent en bas, etc.). On s'intéresse ici au problème de l'Agencement Optimal. Une instance du problème Agencement Optimal est composée :

- d'un entier n.
- de deux séquences de n entiers $x_1, \ldots, x_n$ et $y_1, \ldots, y_n$, représentant respectivement la largeur et la hauteur de chacuns des n blocs.
- d'un entier W représentant la largeur de l'écran.
- d'un entier H.

La question est : « *Existe-t-il un agencement des blocs qui tient en H pixels de haut (ou moins) ?* ».

Agencement Optimal est NP

Dans un premier temps, on veut montrer que ce problème de décision est dans la classe NP.

Q 1. - *1 pt* - Qu'est-ce qui peut faire office de certificat ? Justifiez que sa taille est polynomiale en la taille de l'instance.

Q 2. - *1.5 pts* - Donnez les conditions à vérifier pour garantir qu'un certificat est valide. Démontrez qu'il est possible d'effectuer ces vérifications en temps polynomial, et donc que le problème de décision Agencement Optimal appartient à la classe NP.

Q 3. - *0.5 pt* - Montrez que dans une instance de ce problème de décison,

$$\sum_{i=1}^{n} x_i \cdot y_i > W \cdot H \quad \Longrightarrow \text{ la réponse est « non »}$$

Q 4. - *0.5 pt* - Donnez un exemple où la réponse est non, mais où on a malgré tout :

$$\sum_{i=1}^{n} x_i \cdot y_i \leq W \cdot H$$

Q 5. - *1 pt* - Comment formulerait-on le problème d'optimisation associé ?

Agencement Optimal est NP-dur

On veut maintenant montrer que le problème est NP-dur. Pour cela, on va montrer que le problème Partition se réduit polynomialement en Agencement Optimal. Une instance de Partition est composée de n entiers $a_1, \ldots, a_n$. La question est : « *Peut-on partitionner la liste des n entiers en deux, de telle sorte que les somme des entiers dans les deux moitiés soient égales ?* ».

Par exemple, pour l'instance « 1,2,3,3,3,4,5,6,7,10,12,13,15 », la réponse est « oui », car :

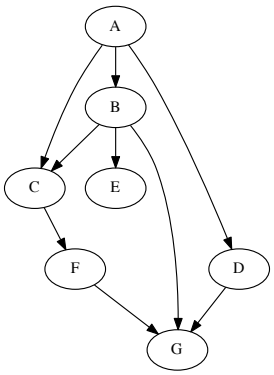
$$1 + 3 + 3 + 5 + 7 + 10 + 13 = 2 + 3 + 4 + 6 + 12 + 15$$

Q 6. - *2.5 pts* - Proposez une réduction polynomiale de Partition en Agencement Optimal et justifiez qu'elle est correcte. **Indice** : réfléchissez à une disposition en deux colonnes.

En fait, il n'est pas réaliste de permettre que les blocs dans chaque page bougent n'importe comment. Les blocs de titre, par exemple, doivent se trouver au-dessus des autres — dans le monde occidental, en tout cas. Pour cela, on considère le problème Agencement Optimal Contraint. C'est la même chose qu'avant, mais en plus les instances contiennent un graphe orienté acyclique dont les sommets sont les blocs. Si une arête relie le bloc u au bloc v, alors :

- Soit le bord supérieur du bloc u se trouve au-dessus de celui du bloc v
- Soit le bord supérieur du bloc u se trouve à la même hauteur que celui du bloc v, mais u doit se trouver à gauche de v.

Par exemple, pour la page web donnée en exemple au début, on considère le graphe :



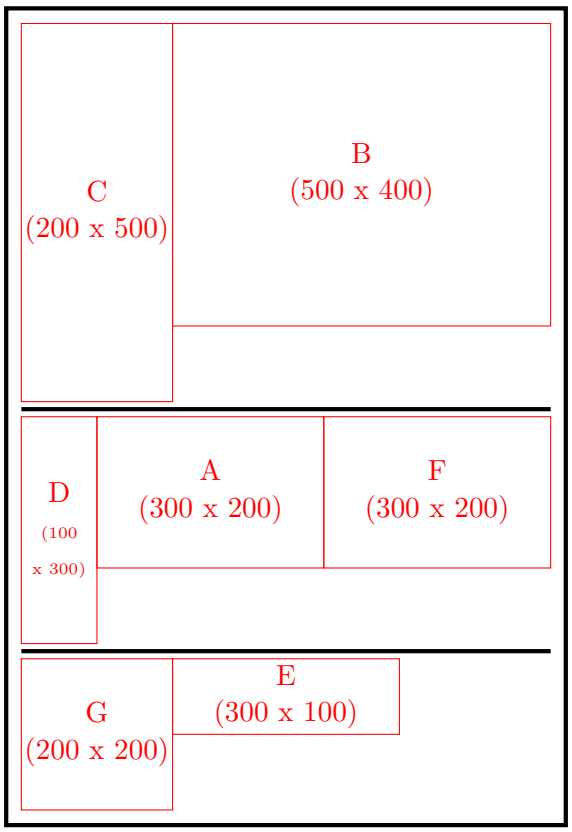
Q 7. - *0.5 pt* - L'exemple d'agencement de gauche respecte-t-il ce graphe ? Et celui de droite ?
Q 8. - *0.5 pt* - Démontrez (en quelques lignes) qu'Agencement Optimal Contraint est NP-complet.

Approximation garantie

On va s'intéresser à une heuristique gloutonne, dans le cas non contraint :

- On trie les rectangles par hauteur décroissante.
- On prend le premier rectangle à traiter, et on le met en haut à gauche.
- Tant que ça tient en largeur, on ajoute les rectangles suivants sur la droite.
- Quand ça ne tient plus, on tire un trait sous les rectangles déjà posés, et on recommence à construire un « niveau » supplémentaire en-dessous.

Sur la figure ci-dessous, on voit cette heuristique à l'oeuvre. Après le placement de C et B, le premier « niveau » est complet, et on passe au niveau d'après (D, A et F).



Q 9. - *1 pt* - Avec quelle complexité peut-on implémenter cette heuristique ? Justifiez.

Etant donnée une instance $\mathcal{I}$ du problème, on note $\mathcal{A}(\mathcal{I})$ le résultat de l'algorithme d'approximation (la hauteur atteinte par l'approximation), et $OPT(\mathcal{I})$ la solution optimale (la plus petite hauteur possible).

Q 10. - *1 pt* - Donnez un exemple où cet algorithme d'approximation donne une solution qui est deux fois (ou environ deux fois) plus haute que la solution optimale.

Soit $y_{\max}$ la hauteur maximale des blocs dans l'instance $\mathcal{I}$. On va maintenant montrer que :

$$\mathcal{A}(\mathcal{I}) \leq y_{\max} + 2 \cdot OPT(\mathcal{I})$$

On considère les différents « niveaux » produits par l'algorithme d'approximation, dans l'ordre où ils sont produits par l'algorithme. On note X_i et Y_i la largeur et la hauteur du rectangle *de gauche* du niveau i, ainsi que $\widetilde{X}_i$ la largeur *totale* du niveau i (la somme des largeurs des blocs du niveau i), A_i la surface du niveau i (la somme des surfaces des blocs du niveau i) et k le nombre de niveaux.

Nous avons détaillé le raisonnement en questions (plus ou moins indépendantes). Par conséquent, les questions suivantes admettent des réponses très simples, en une ligne ou deux.

Q 11. - *0.25 pt* - Justifiez que $Y_1 \geq Y_2 \geq \cdots \geq Y_k$.

Q 12. - *0.25 pt* - Justifiez que $A_i \geq \widetilde{X}_i \cdot Y_{i+1}$.

Q 13. - *0.25 pt* - Justifiez que $A_i + A_{i+1} \geq Y_{i+1}(\widetilde{X}_i + X_{i+1})$.

Q 14. - *0.25 pt* - Justifiez que $A_i + A_{i+1} > W \cdot Y_{i+1}$.

Q 15. - *0.25 pt* - Justifiez que $\mathcal{A}(\mathcal{I}) = \sum_{i=1}^{k} Y_i$.

Q 16. - *0.25 pt* - Justifiez que $\mathcal{A}(\mathcal{I}) \leq Y_1 + \frac{1}{W} \sum_{i=1}^{k-1} (A_i + A_{i+1})$.

Q 17. - *0.25 pt* - Justifiez que $\frac{1}{W} \sum_{i=1}^{k} A_i \leq OPT(\mathcal{I})$

Q 18. - *0.25 pt* - Concluez que $\mathcal{A}(\mathcal{I}) \leq y_{\max} + 2 \cdot OPT(\mathcal{I})$.