Algorithmique
Examen de deuxième session
21 juin 2018 12h00–15h00

Documentation autorisée : deux feuilles A4 recto verso manuscrites. Tout autre document ou appareil est interdit. **Les téléphones portables doivent être éteints et rangés dans votre sac, et vous devez poser sacs et vestes à l'avant de la salle avant de commencer.**

Indications : le barème est donné à titre indicatif seulement. Sauf indication contraire, justifiez vos réponses et expliquez brièvement vos récurrences et vos algorithmes. Une partie de la note portera sur la clarté de présentation de vos réponses.

Exercice 1 : (5 points)

Le problème des occurrences consiste à compter les occurrences des mots dans un texte (qu'on représente ici comme une liste de mots) :

En entrée : Une liste de N mots $W = w_1, \ldots, w_N$

En sortie : Une liste de paires (v_i, c_i) où les mots v_i sont les mots distincts de la liste W et c_i est le nombre d'occurrences de v_i dans W.

On souhaite écrire un algorithme de type diviser-pour-régner pour ce problème. On peut imaginer plusieurs variantes et on souhaite savoir laquelle serait la meilleure. On suppose ici que tous les mots sont composés d'au plus 30 caractères et que les comparaisons peuvent être faites à coût unitaire.

Algorithme DPR1 Diviser la liste W en deux listes de taille égale. Résoudre le problème récursivement sur les deux sous-listes. Fusionner les deux résultats retournés par les appels récursifs. Dans cette variante, les deux sous-résultats sont retournés dans un ordre quelconque.

Algorithme DPR2 Diviser la liste W en deux listes de taille égale. Résoudre le problème récursivement sur les deux sous-listes. Fusionner les deux résultats retournés par les appels récursifs en prenant soin de retourner le résultat dans l'ordre alphabétique des mots.

1. Donner un algorithme `fusion1(L1,L2)` qui implémente la procédure de fusion de l'algorithme DPR1.
2. Donner la complexité asymptotique en pire cas $f_1(m)$ de l'algorithme `fusion1`, où m est le nombre total de mots dans les deux listes.
3. Écrire la récurrence de complexité en temps $T_1(N)$ de l'algorithme DPR1, implémenté à l'aide de l'algorithme de fusion `fusion1`, où N est le nombre de mots dans la liste W donnée en entrée.
4. Résoudre la récurrence $T_1(N)$.
5. Donner un algorithme `fusion2(L1,L2)` qui implémente la procédure de fusion de l'algorithme DPR2.
6. Donner la complexité asymptotique en pire cas $f_2(m)$ de l'algorithme `fusion2`.

7. Écrire la récurrence de complexité en temps $T_2(N)$ de l'algorithme DPR2, implémenté à l'aide de l'algorithme de fusion `fusion2`
8. Résoudre la récurrence $T_2(N)$.

Exercice 2 : (6 points)

On cherche dans un tableau d'entiers T de taille $m \times n$, un sous-tableau carré de taille maximale contenant uniquement des zéros.
Un sous-tableau carré est déterminé par trois entiers (i, j, t) avec $0 \leqslant i < m$, $0 \leqslant j < n$, et $0 \leqslant t < \min(m, n)$ interprété comme suit : la valeur T[u][v] est dans le sous-tableau carré (i, j, t) si et seulement si $i \leqslant u < i+t$ et $j \leqslant v < j+t$. On dit qu'un sous-tableau carré déterminé par les indices i, j, t commence aux coordonnées (i, j), et sa taille est $t \times t$.
Par exemple, le tableau suivant contient un sous-tableau carré composé uniquement de zéros de taille 3×3 qui commence aux coordonnées $(2, 1)$:

	$j=0$	$j=1$	$j=2$	$j=3$	$j=5$
$i=0$	0	1	1	0	1
$i=1$	1	0	1	1	0
$i=2$	0	0	0	0	1
$i=3$	0	0	0	0	1
$i=4$	1	0	0	0	1
$i=5$	0	1	1	0	1

1. Donner une récurrence qui permet de calculer le sous-tableau carré de taille maximale contenant uniquement des zéros. *Indication :* vous pouvez écrire une récurrence $R(i, j)$ qui exprime, pour chaque couple i, j dans le tableau, la taille du sous-tableau carré contenant uniquement des zéros qui commence aux coordonnées (i, j). Vous pouvez utiliser trois valeurs précédemment calculées pour calculer la valeur de $R(i, j)$.
2. Expliquer comment cette récurrence peut donner lieu à un algorithme de type programmation dynamique en renseignant les questions suivantes. (a) Donner les dimensions du ou des tableaux, (b) Donner le schéma des dépendances sous forme de diagramme, (c) Dire dans quel ordre le tableau doit être rempli, (d) Préciser les conditions de bord, (e) Dire comment trouver la valeur de la solution optimale lorsque toutes les valeurs de la récurrence ont été calculées.

Exercice 3 : (5 points)

Chaque mois, un propriétaire d'une chambre d'hôte reçoit une liste de demandes de location $L = (A_1, D_1), \ldots, (A_n, D_n)$ qui consistent en une date d'arrivée A_i et d'une date de départ D_i. Le propriétaire doit choisir un sous-ensemble de demandes qui maximise le nombre de jours où sa chambre est occupée.
On propose plusieurs algorithmes gloutons pour ce problème. Pour chaque algorithme, dire s'il est optimal ou non. S'il n'est pas optimal, donner un contre-exemple simple. S'il est optimal, donner une preuve d'optimalité.

Algorithme G1 Ordonner les demandes de réservation par ordre croissant de durée. (La durée d'une demande de réservation est le nombre de nuits passées, soit $D_i - A_i$.) Parcourir la liste dans cet ordre et sélectionner la demande si les dates ne chevauchent pas une autre demande de réservation déjà sélectionnée.

Algorithme G2 Ordonner les demandes de réservation par ordre décroissant de durée. Parcourir la liste dans cet ordre et sélectionner la demande si les dates ne chevauchent pas une autre demande de réservation déjà sélectionnée.

Algorithme G2 Ordonner les demandes de réservation par ordre croissant de date d'arrivée. Parcourir la liste dans cet ordre et sélectionner la demande si les dates ne chevauchent pas une autre demande de réservation déjà sélectionnée.

Algorithme G4 Ordonner les demandes de réservation par ordre décroissant de date d'arrivée. Parcourir la liste dans cet ordre et sélectionner la demande si les dates ne chevauchent pas une autre demande de réservation déjà sélectionnée.

Exercice 4 : (5 points)

Le problème D-CLIQUE est défini comme suit :

En entrée : un graphe G

En sortie : `Vrai` s'il existe une clique de taille au moins $\lfloor n/2 \rfloor$, où n est le nombre de sommets de G, `Faux` sinon.

Montrer que le problème D-CLIQUE est NP-complet.

Indication : On rappelle que le problème NP-complet CLIQUE est défini comme suit.

En entrée : un graphe G et un entier c

En sortie : `Vrai` s'il existe une clique de taille c dans G, `Faux` sinon.

Pour la réduction, penser à ajouter s sommets isolés et/ou r sommets reliés à tous les sommets du graphe G.