Algorithmique
Examen de première session
11 janvier 2018 8h30–11h30

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)

Un revendeur cherche à maximiser son profit en achetant un item à bas prix et en le revendant à prix élevé. Le problème du plus grand profit consiste à déterminer, étant donné la liste des prix de vente sur une période, à quel jour i acheter et à quel jour $j \geqslant i$ revendre afin de maximiser son profit `prix[j]-prix[i]`. (Bien entendu, on doit acheter avant de vendre.)
Par exemple, si la liste de prix est `[9, 2, 4, 6, 1, 4, 2, 8, 3]` on devrait acheter au jour 4 lorsque le prix est 1 et revendre au jour 7 lorsque le prix est 8 pour un profit de 7 unités. (Les indices d'une liste commencent à 0.)

1. Donner la complexité d'un algorithme qui consiste à tester toutes les paires d'indices (i, j) (avec $i \leqslant j$) et retourner la paire qui réalise le profit maximum.
2. Donner une récurrence structurelle qui permet de résoudre ce problème avec un algorithme de type diviser-pour-régner.
3. Donner la récurrence de complexité de l'algorithme qui implémente votre récurrence structurelle. Justifier brièvement.
4. Résoudre la récurrence de complexité et comparer à la complexité de l'algorithme de la question 1.

Exercice 2 : (6 points)

Un analyste financier souhaite extraire à chaque mois la période (jour de début, jour de fin) où se produit le plus grande croissance sur un titre. Les données sont présentées sous forme d'une liste d'entiers qui représente pour chaque jour, la variation du cours du titre par rapport à la veille, en croissance si le nombre est positif ou en décroissance si le nombre est négatif. La croissance nette sur un intervalle est donc la somme de toutes les valeurs sur cet intervalle. Par exemple, si la liste est `[-2, 2, -4, 3, 6, -8, 4]`, la croissance est 7 sur l'intervalle de valeurs `[2, -4, 3, 6]` et 5 sur l'intervalle `[3, 6, -8, 4]`.

1. Donner une récurrence qui permet de trouver la valeur de la plus grande croissance sur un intervalle de jours consécutifs. Faire le calcul pour la liste `[-2, 2, -4, 3, 6, -8, 4]`.
   Indication : penser à une récurrence qui exprime, pour chaque indice i du tableau, la croissance maximale qu'on peut obtenir sur un intervalle commençant à l'indice i.

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.
3. Décrire quelle information supplémentaire il faut stocker pour obtenir en sortie non seulement la valeur de la solution optimale, mais aussi l'intervalle où se produit l'optimum.

Exercice 3 : (5 points)

Etant donné un ensemble de tâches à réaliser sur un unique processeur, on souhaite trouver un ordonnancement de toutes les tâches qui minimise le retard. (Le retard dans un ordonnancement est le retard maximum, pris sur l'ensemble des tâches.) Pour chaque tâche j, on connaît sa durée d'exécution $duree_j$, et une échéance fin_j, après laquelle elle est considérée en retard. Si une tâche j se termine avant fin_j, elle n'a aucun retard mais si elle se termine à un temps $t > fin_j$, elle encourt un retard de $t - fin_j$ unités de temps. On propose plusieurs algorithmes gloutons pour ce problème :

OMR1 Ordonner les tâches par ordre de durée croissant.

OMR2 Ordonner les tâches par ordre d'*urgence* croissant. L'urgence d'une tâche j est $fin_j - duree_j$, c'est-à-dire le moment où on doit commencer si on ne veut pas que cette tâche soit en retard.

OMR3 Ordonner les tâches par ordre d'échéance croissante.

1. Montrer que OMR1 n'est pas optimal.
2. Montrer que OMR2 n'est pas optimal.
3. Montrer que OMR3 est optimal. *Indication :* considérer deux tâches consécutives.

Exercice 4 : (4 points)

Le problème MOITIÉ est défini comme suit :

En entrée : un ensemble d'entiers positifs ou nuls $Y = \{y_1, \cdots, y_m\}$ et un entier s.

En sortie : `Vrai` s'il existe un sous-ensemble $P \subseteq Y$ de taille $\lfloor m/2 \rfloor$ dont la somme des éléments vaut s, `Faux` sinon.

Montrer que le problème MOITIÉ est NP-complet.

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

En entrée : un ensemble d'entiers positifs ou nuls $X = \{x_1, \cdots, x_n\}$ et un entier t.

En sortie : `Vrai` s'il existe un sous-ensemble $S \subseteq X$ tel que la somme des éléments de S est égale à t, `Faux` sinon.