Algorithmique
Examen
jeudi 10 janvier 2019 8h30–11h30

Documentation autorisée : une feuille A4 recto verso manuscrite. Tout autre document ou appareil est interdit. **Les téléphones portables doivent être éteints et rangés dans votre sac, et les sacs et vestes posés à l'avant de la salle avant de commencer.** **Indications** : le barème est donné à titre indicatif seulement. Sauf indication contraire, vous devez justifier vos réponses et expliquer vos algorithmes. Les algorithmes doivent être aussi efficaces que possible et présentés de façon lisible et claire. Cela veut dire que les noms de variables doivent être bien choisis pour représenter ce qu'elles sont censées contenir et que les principales étapes doivent être accompagnées d'explications. Une partie de la note portera sur la clarté de présentation de vos réponses.

Exercice 1 : (6 points)

Le problème des DEUXVALISES est défini comme suit.

En entrée : Des poids entiers positifs $p_1, \dots p_n$ de n items à répartir dans deux valises, et des capacités de deux valises C_1, C_2.

En sortie : `Vrai` si on peut répartir les items $\{1, \dots, n\}$ dans les deux valises sans dépasser leur capacité, `Faux` sinon.

1. Donner une relation de récurrence pour le problème des DEUXVALISES. *Indication :* considérer $V(i, X, Y)$ dont la valeur est `Vrai` si on peut ranger les items $\{i, \dots, n\}$ dans des valises de capacités X et Y.
2. Donner un algorithme de type backtracking qui se base sur cette relation de récurrence. Quelle est la complexité de cet algorithme en fonction de C_1, C_2, n ?
3. Quelles sont les valeurs que peuvent prendre i, X et Y au cours de l'exécution de l'algorithme de la question 2 s'il y a n items et les capacités initiales des valises sont C_1 et C_2 ? (Donner des intervalles de valeurs possibles pour X et Y, et un intervalle de valeurs possibles pour i.)
4. Si on devait donner un algorithme de type programmation dynamique pour ce problème en utilisant la même récurrence, quelle en serait la complexité en fonction de C_1, C_2, n ?

Exercice 2 : (4 points)

Montrer que le problème des DEUXVALISES décrit à l'exercice 1 est NP complet.
Indication : le problème SUBSETSUM est NP-complet.

Le problème SUBSETSUM :

En entrée : Une liste de valeurs positives $L = [x_1, \dots, x_n]$ et une valeur cible T.

En sortie : `Vrai` s'il existe un sous ensemble de valeurs dans L de somme T, `Faux` sinon.

Exercice 3 : (6 points)

Un voyageur souhaite aller de Paris à Papeete (Tahiti) en 10 escales, en minimisant les coûts de transport et de logement. Chaque jour il prendra un vol et dormira une nuit à l'hôtel. Afin de résoudre le problème en toute généralité, on le modélise à l'aide d'un graphe dont les sommets représentent les villes, et les arêtes représentent les vols disponibles. On suppose qu'il y a n villes numérotées de 1 à n. Pour la ville i, le coût de l'hôtel est $H[i]$ et pour le trajet de la ville i à la ville j, le prix du vol est $V[i][j]$. (S'il n'y a pas de vol, le prix est $+\infty$.) En entrée, on précisera une ville de départ *Dep* et une ville d'arrivée *Arr*, ainsi que le nombre d'escales T.
Le but de cet exercice est d'obtenir un algorithme de programmation dynamique pour trouver l'itinéraire de meilleur coût de la ville *Dep* à la ville *Arr* dans un graphe de n villes en faisant T escales, en répondant aux questions ci-dessous.

1. Donner une relation de récurrence pour ce problème de la forme suivante : $M(k,t)$ est le coût minimal pour aller de la ville *Dep* à la ville k en t escales.
   Indication : pour se rendre à la ville k en t escales, on devra faire escale à une ville ℓ atteinte en $t-1$ escales.
2. En supposant que toutes les valeurs utiles aient déjà été calculées, combien de temps (qu'on notera $c(n)$) faut-il pour calculer une nouvelle valeur de la relation de récurrence (c'est à dire pour remplir une case du tableau) ?
3. Donner un algorithme de programmation dynamique basé sur la relation de récurrence donnée à la question 1.
4. Donner la complexité de votre algorithme, d'abord en fonction de T, n et $c(n)$, puis en développant l'expression $c(n)$ obtenue à la question 2 pour obtenir une expression en fonction de n et T seulement.

Exercice 4 : (4 points)

Le problème de BINPACKING est défini comme suit.

En entrée : Une liste de poids $L = [x_1, \ldots, x_n]$, $0 < x_i \leqslant 1$

En sortie : Le plus petit nombre B de boîtes de capacité 1 dans lequel il est possible de répartir tous les poids.

1. On propose les algorithmes gloutons suivants pour BINPACKING. Pour chacun, donner un exemple où il n'est pas optimal, et comparer avec la valeur optimale.
   Bonus : trouver un exemple qui approche le plus possible le facteur d'approximation, donné entre parenthèses.
   a. FIRSTFIT : Prendre les poids dans l'ordre donné. Mettre le poids dans la première boîte dans lequel il rentre. (Cet algorithme est une 17/10 approximation.)
   b. FIRSTFITINCREASING : Prendre les poids en ordre croissant. Mettre le poids dans la première boîte dans lequel il rentre. (Cet algorithme est une 17/10 approximation.)
   c. FIRSTFITDECREASING : Prendre les poids en ordre décroissant. Mettre le poids dans la première boîte dans lequel il rentre. (Cet algorithme est une 3/2 approximation.)
2. Montrer que FIRSTFIT est une 2-approximation.