École polytechnique — Majeure informatique
Promotion 2008

Cours *Conception et analyse d'algorithmes*
TD n° 2 – Programmation récursive et programmation dynamique
22 septembre 2010

1. Sous-arbre de poids maximal

On considère un arbre A et une fonction α qui associe un poids $\alpha(v)$ (positif ou négatif) à chaque sommet v de A. On considère les *sous-arbres* de A au sens des graphes, c'est-à-dire les sous-ensembles X de sommets tels que pour tout sommets $u, v \in X$ le chemin de u à v dans l'arbre A ne passe que par des sommets de X. Donner un algorithme qui détermine en temps linéaire un sous-arbre de A de poids maximal (le poids d'un sous-arbre étant la somme des poids des sommets qu'il contient).

2. Reconstruction de texte

On se donne un texte dont on pense qu'il a été obtenu à partir d'un document en français en supprimant tous les espaces et symboles de ponctuation ("longtemps-jemesuislevédebonneheure..."). On souhaite reconstruire le document à l'aide d'un dictionnaire, auquel on accède par une fonction booleenne `dict()` qui renvoit pour toute chaîne s

$$\texttt{dict}(s) = \begin{cases} \texttt{vrai} & \text{si } w \text{ est un mot du dictionnaire} \\ \texttt{faux} & \text{sinon.} \end{cases}$$

1. Décrire un algorithme qui décide si le texte peut être décomposé en mots du dictionnaire. L'algorithme devrait être de complexité $O(n^2)$ (un appel à la fonction `dict` étant compté comme une opération élémentaire).

2. Dans le cas où le texte peut être décomposé, donner un algorithme qui propose une reconstitution possible. Quelle est la complexité de votre algorithme, et l'espace mémoire dont il a besoin ?

3. Algorithme de Viterbi

On peut se servir de la programmation dynamique sur un graphe orienté de reconnaissance vocale $G = (S, A)$. Chaque arc (u, v) est étiqueté avec un phonème $s(u, v)$ tiré d'un ensemble fini P. Ce graphe est un modèle formel d'une personne parlant un langage restreint. Chaque chemin du graphe partant d'un sommet distingué v_0 de S correspond à une séquence de sons pouvant être produite par le modèle. L'étiquette d'un chemin est définie comme étant la concaténation des étiquettes des arcs du chemin.

i. Décrire un algorithme efficace qui, étant donné un graphe G étiqueté, un sommet distingué v_0 et une séquence $s = s_1 \dots s_n$ de caractères de P, retourne un chemin de G qui commence en v_0 et d'étiquette s si un tel chemin existe.

ii. On associe maintenant à chaque arc (u, v) une probabilité de transition $p(u, v)$. La somme des probabilités des arcs sortant d'un sommet vaut 1. La probabilité d'un chemin est le produit des probabilités de ses arcs. Étendre la réponse de la question précédente pour que le chemin retourné soit un chemin *le plus probable* partant de v_0 et d'étiquette s. Quel est le temps d'exécution de l'algorithme ?

4. Ordonnancement pour minimiser le nombre de tâches en retard

On considère de nouveau un problème d'ordonnancement de tâches, en supposant cette fois que les tâches T_i $(1 \leqslant i \leqslant n)$ ont une durée d_i. Pour chaque tâche sont fixés une date d'échéance e_i et une pénalité p_i à acquitter quand la tâche n'est pas terminée avant la date e_i. On veut trouver un ordonnancement des tâches sur une machine unique qui minimise la somme des pénalités de retard.

i. Montrer que lorsqu'on connaît l'ensemble des tâches qui sont exécutées sans retard, il est facile de trouver un ordonnancement optimal.

ii. Montrer qu'on peut déterminer l'ensemble des tâches à exécuter sans retard par programmation dynamique. Les tâches étant classées selon leur date d'échéance, on considèrera pour cela la durée minimale d'un ordonnancement sans retard d'un sous-ensemble de tâches inclus dans $\{T_1, \cdots, T_j\}$ et de pénalité totale supérieure ou égale à p.

5. Problème du voyageur de commerce

Le *problème du voyageur de commerce* consiste en la donnée d'un graphe complet non-orienté à n sommets, numérotés de 1 à n (et donc $n(n-1)/2$ arêtes) ; à chaque arête est associée une distance. Le voyageur de commerce réside au sommet 1 du graphe, et cherche à organiser sa tournée de façon à partir de 1, revenir en 1, passer une fois et une seule par chacun des autres sommets du graphe, et ce en parcourant la distance minimale.

i. Quelle est le nombre d'étapes de calcul à effectuer pour une énumération exhaustive des tournées possibles ?

ii. Soit $S \subset \{2, 3, \ldots, n\}$ et $k \in S$. Soit $C(S, k)$ la distance minimale pour aller de 1 à k en passant une fois et une seule par tous les sommets de S. Exprimer $C(S, k)$ en fonction des $C(S - \{k\}, l)$, pour $l \in S - \{k\}$. Quelle technique utiliser pour exploiter cette relation (justifier le choix) ? Écrire le pseudo-code correspondant. Quel est le nombre d'étapes de calcul par cette méthode ? Comparer avec **i**.