École polytechnique — Majeure informatique
Promotion 2001

Cours de conception et analyse d'algorithmes

PC 1 – algorithmes gloutons, programmation dynamique

1. Sac à dos.

On souhaite emporter en voyage certains objets parmi n ; l'objet i a poids p_i et valeur a_i. On est limité par le poids maximum total P des objets à emporter et on cherche donc un sous-ensemble d'objets de valeur totale maximale parmi ceux dont la somme des poids est inférieure ou égale à P. Considérer les tentatives d'algorithmes suivants, et pour chacun d'eux, montrer par un exemple que cet algorithme ne donne pas toujours l'optimum.

i. Classer les objets par valeurs décroissantes puis considérer les objets un par un dans cet ordre ; on emporte ceux dont le poids ne fait pas dépasser le poids total P autorisé.

ii. Classer les objets par poids croissants et poursuivre comme plus haut.

iii. Classer les objets suivant le rapport (valeur/ poids) et poursuivre comme plus haut.

2. Implantation de l'algorithme de Kruskal.

Pour l'algorithme de Kruskal, la gestion des sous-arbres est essentielle pour une bonne complexité. On se réduit à l'étude de la structure de données abstraite *Union-find*. Le problème est de gérer une partition d'un ensemble E de taille n en k sous-ensembles E_i disjoints. A chaque partie E_i correspond un élément particulier unique appelé le *représentant* de E_i. On souhaite pouvoir effectuer deux types d'opérations :

- la recherche : $R(x)$ doit renvoyer le représentant de l'ensemble E_k auquel appartient x. (Cette opération permet de décider si deux éléments appartiennent au même sous-ensemble de la partition.)
- l'union : $U(i, j)$ doit, étant donnés les représentants de E_i et de E_j, remplacer une partition $\mathcal{E} = \{E_1, \dots, E_n\}$ par la partition $\mathcal{E} - E_i - E_j + (E_i \cup E_j)$.

i. Donner une structure de données très simple pour laquelle l'union a coût $O(1)$ et la recherche a coût $O(n)$.

ii. Donner une structure de données pour laquelle la recherche a coût $O(1)$, au prix d'un accroissement du coût de l'union.

iii. On propose maintenant d'implanter une partie sous forme d'arbre dans lequel chaque nœud est étiqueté par un élément de cet ensemble et pointe vers son père, la racine pointant vers elle-même. L'élément $R(x)$ renvoyé par l'opération de recherche est la racine de l'arbre.

Décrire l'implantation des opérations de recherche et d'union. Montrer que l'union s'effectue encore en temps constant.

Montrer que l'on peut de plus maintenir un entier *rang* associé à chaque nœud, égal, pour les racines des arbres, à la hauteur de l'arbre enraciné en ce noeud.

Lors de l'opération d'union, on peut choisir comme racine de $E_i \cup E_j$ celle des deux racines de E_i et de E_j qui a le rang le plus élevé. Montrer qu'alors un arbre de hauteur h contient au moins 2^h noeuds. En déduire que pour cette implantation, l'union a coût $O(\log n)$.

iv. Quelle est la complexité de l'algorithme de Kruskal ainsi implanté ?

3. Triangle de Pascal

On rappelle que le coefficient binomial C_n^k est le nombre de sous-ensembles à k éléments d'un ensemble à n éléments. Proposer un algorithme de calcul de C_n^k basé sur la programmation dynamique.

4. Insertion d'espaces dans un texte

Pour chiffrer un texte (par exemple par substitution lettre par lettre), on commence par en supprimer tous les espaces. Le destinataire, une fois le texte déchiffré, se retrouve donc avec un bloc de texte dans lequel il faut réinsérer des espaces.

On suppose que l'on dispose d'un dictionnaire contenant la liste de tous les mots valides (y compris les pluriels, féminins, formes conjuguées des verbes, etc), accessible par une fonction booléenne *est-dans-dico(mot m)*. Proposer un algorithme basé sur la programmation dynamique pour prendre le bloc de texte donné en entrée et décider s'il existe une insertion d'espaces découpant le bloc en une suite de mots valides. On écrira le pseudo-code correspondant. Quelle est la complexité de l'algorithme ? Le compléter pour que de plus il calcule un découpage lorsqu'il en existe un.

5. Organisation d'un pot

Une PDG souhaite organiser un pot pour fêter la décoration qui vient de lui être décernée. Les relations hiérarchiques dans l'entreprise ont une structure d'arbre dont la PDG est la racine. Elle souhaite qu'un employé ne soit pas invité en même temps que son supérieur immédiat. Qui plus est, à chaque employé est associée une “note de convivialité". Comment prévoir la liste des invités de façon à maximiser la somme des notes de convivialité ? Comment assurer que la PDG sera invitée ?

6. Algorithme de Viterbi

Considèrons un graphe orienté de reconnaissance vocale $G = (S, A)$ modélisant une personne parlant un langage restreint. Chaque arc est étiqueté par un phonème tiré d'un ensemble fini. Chaque chemin du graphe partant d'un sommet distingué v_0 correspond à une séquence de sons. L'étiquette d'un chemin est définie comme la concaténation des étiquettes de ses arcs.

i. Proposer un algorithme qui, étant donné une séquence $s = s_1 \dots s_n$ de phonèmes, 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 $p(u, v)$ représentant les chances de passer par l'arc (u, v) à partir du sommet u. 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* parmi les chemins partant de v_0 et d'étiquette s. Quel est le temps d'exécution de l'algorithme ?