Programmation avancée

Répétition 9: Algorithmes gloutons

Jean-Michel Begon

14 décembre 2018

1 Approximation gloutonne

Proposez une approximation gloutonne, ainsi qu'un contre-exemple démontrant sa sous-optimalité pour les problèmes suivants :

1. La chasse au trésor.
2. Découpage de la tige d'acier.
3. L'ordre des multiplication de matrices.

2 Arbre couvrant minimum

Soit le graphe valué $G = (V, E, W)$, où V est l'ensemble des nœuds (*vertex*) du graphe, $E \subseteq V \times V$ est l'ensemble des arêtes (*edge*) de G et $W : E \to \mathbb{R}_+$ est la fonction de poids de chaque arête. Par exemple, les nœuds représentent des routeurs dans un réseau informatique et le poids correspond à la capacité du lien. On désire réaliser un arbre couvrant minimum (*Minimum spanning tree*), c'est-à-dire un arbre (graphe acyclique connexe) qui est tel que la somme des poids des branches est minimum. Proposez un algorithme *greedy* pour solutionner ce problème.

3 La course de Julien (adapté de CLRS, 16.2-4)

Julien participe à une course à pieds qui consiste à relier une ville A à une ville B. Etant donné la contenance de son bidon, il sait qu'il peut parcourir m mètres sans tomber à cours d'eau. Il a également pu obtenir des organisateurs du marathon la liste des endroits où il pourra remplir son bidon et les distances entre ces endroits.

Julien aimerait minimiser le nombre d'arrêts qu'il devra effectuer lors de sa course pour remplir son bidon. Donnez une méthode efficace pour déterminer quels arrêts il devrait faire. Montrez que votre stratégie donne une solution optimale et discutez sa complexité.

4 L'ordonnancement

L'ordonnancement des tâches est un problème fréquent en informatique : on dispose d'un ensemble $S = \{s_1, s_2, ..., s_n\}$ de n tâches qu'on doit ordonner. A chaque tâche s_i est associé un temps d'exécution t_i. Une fois l'ordre des tâches fixé, on peut également associer à chaque tâche un temps de complétion (ou temps de réponse), c'est-à-dire, le temps qui s'est écoulé jusqu'à la fin de la tâche.

Par exemple, si on dispose des tâches s_1 et s_2 dont les temps d'exécution respectifs sont $t_1 = 3$ et $t_2 = 5$ et si on exécute les tâches dans l'ordre $< s_1, s_2 >$, alors les temps de complétion seront :

— $C_1 = t_1 = 3$
— $C_2 = C_1 + t_2 = 3 + 5 = 8$

En outre, la somme des temps de complétion sera $C_\Sigma = \sum_i C_i = C_1 + C_2 = 11$.

On souhaite disposer d'un algorithme d'ordonnancement des tâches qui minimise la somme des temps de complétion (C_Σ).

1. Quelle serait la complexité d'un algorithme *brute-force* pour résoudre ce problème ?
2. Est-il possible de faire mieux ?(*i.e.* le problème dispose-t-il de la propriété de sous-structure optimale ?)
3. Le cas échéant, proposez un algorithme efficace pour résoudre ce problème.

5 Intervalles unitaires (CLRS, 16.2-5)

Etant donné un ensemble $x_1, x_2, ..., x_n$ de points sur l'axe réel, donner un algorithme efficace pour déterminer le plus petit ensemble d'intervalles fermés de longueur unitaire qui contient tous les points. Montrez que votre algorithme est correct.

6 Je vais au marché

Vous devez acheter n produits. Le prix initial de chaque produit est 1 euro mais chaque jour le prix du $i$ème produit augmente d'un facteur $f(i)$. Développez un algorithme efficace pour trouver l'ordre optimal dans lequel les produits devraient être acheté, en supposant que vous pouvez acheter au plus un produit chaque jour.

Bonus – La peinture (adapté de l'IEEExtreme 2016)

Vous souhaitez mettre en couleur une mosaïque. Comme vous êtes méthodiques, vous avez déjà acheté tous les pigments nécessaire et vous avez décidé de peindre la toile de haut en bas et droite à gauche. Vous connaissez donc exactement la suite de couleurs à utiliser. Vous disposez de deux pinceaux et souhaitez minimiser le nombre de fois que vous devez nettoyer votre pinceau. Quelle stratégie allez-vous adapter ?