École polytechnique — Majeure Informatique
Promotion 2000

Cours de Conception et analyse d'algorithmes
Examen du mercredi 26 mars 2003. 2 heures
Documents autorisés : poly et notes de cours

Les exercices sont donnés dans l'ordre du cours et non pas spécialement par ordre de difficulté. Ils sont indépendants les uns des autres.

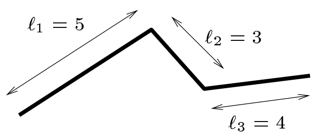
On demande aux étudiants d'accorder une attention particulière à une rédaction soignée, claire et détaillée, qui sera un critère important dans l'évaluation.

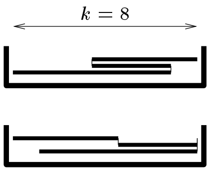
1. Algorithme glouton.

On considère un ensemble d'intervalles, où le i-ième intervalle est $[s_i, t_i[$. Proposer un algorithme de type glouton pour donner une couleur à chaque intervalle de façon que deux intervalles d'intersection non vide soient toujours de couleurs diférentes, en utilisant le nombre minimum de couleurs distinctes au total. Dans cet exercice, une description en français suffira (on ne vous demande pas d'écrire le pseudo-code). Démontrer que votre algorithme est correct. Analyser sa complexité.

2. Programmation dynamique.

On considère un mètre de menuisier, dont les n segments ont longueurs $\ell_1, \ell_2, \ldots, \ell_n$, qui dans cet exercice peuvent être des entiers quelconques (contrairement aux mètres habituels, dont tous les segments ont même longueur), mais tels que $\ell_1 + \ell_2 + \ldots + \ell_n = L \leq 10n$.





On souhaite ranger ce mètre dans une boîte de longueur entière k. Pour cela, il faut replier le mètre de façon que sa longueur totale, une fois replié, soit inférieure ou égale à k. Pour tout i, le i-ième segment peut être replié dans le même sens que le segment $i-1$ ou dans le sens contraire.

Proposer un algorithme basé sur la programmation dynamique et qui permette, étant donné k, n, et $\ell_1, \ldots, \ell_n$ tels que $\sum_i \ell_i \leq 10n$, de *décider* en temps polynomial s'il est possible de replier le mètre de façon à ce qu'il tienne dans la boîte. On souhaite ici que l'algorithme soit suffisamment détaillé (pseudo-code par exemple).

3. Flots.

n personnes $P = \{p_1, \ldots, p_n\}$ partagent leurs véhicules pour aller travailler pendant m jours. Le jour i, un sous-ensemble S_i de P utilisent un véhicule commun pour aller travailler. Le conducteur ce jour-là sera un élément de S_i. Etant donné P et $S_1, \ldots S_m$, le problème est de choisir pour chaque j le conducteur du jour j de façon que la répartition soit équitable, c'est-à-dire qu'une personne i ne conduise pas plus de $\lceil \sum_{j:i \in S_j} \frac{1}{|S_j|} \rceil$ jours au total. Par exemple, s'il y a quatre personnes et deux jours $S_1 = \{p_1, p_2\}$ et $S_2 = \{p_1, p_3, p_4\}$, la personne p_1 doit conduire au maximum $\lceil 1/2 + 1/3 \rceil = 1$ des deux jours.

Modéliser ce problème comme un problème de flot maximum.

Montrer qu'il existe toujours une répartition équitable.

4. NP-complétude.

On considère le problème $4SAT$ qui prend en entrée une formule booléenne sous forme normale conjonctive à m clauses et n variables, où chaque clause est la disjonction d'exactement 4 littéraux, et doit décider si la formule est satisfaisable. Montrer avec soin, en utilisant la complétude de $3SAT$, que ce problème est NP-complet.

5. Algorithme d'approximation.

Le problème de la satisfaction de personnes se présente comme suit. L'entrée consiste en n objets dont les valeurs $a_1, \dots a_n$ sont des nombres rationnels strictement positifs. Il s'agit d'allouer ces objets à des personnes de façon à maximiser le nombre de personnes satisfaites. Une personne est satisfaite si et seulement si la valeur totale des objets qu'elle reçoit est supérieure ou égale à 1.

On propose l'algorithme suivant.

Initialisation : $p \leftarrow 1$

Boucle : pour i allant de 1 à n faire

- donner l'objet i à la personne p
- si elle est maintenant satisfaite, alors $p \leftarrow p + 1$.

Montrer que le nombre de personnes satisfaites par cet algorithme est d'au moins $OPT/2$, où OPT dénote le nombre maximum de personnes pouvant être satisfaites par $\{a_1, \dots a_n\}$. On pourra commencer par envisager le cas particulier où tous les a_i sont inférieurs ou égaux à 1.

6. Algorithme probabiliste.

(i). On suppose qu'on a accès à une source de bits aléatoires indépendants, c'est-à-dire à une suite de variables aléatoires X_i indépendantes, où chaque X_i vaut 0 ou 1 avec probabilités $1/2, 1/2$.

Proposer un algorithme pour générer une suite de n variables aléatoires Z_j qui soient indépendantes et égales à 0,1 ou 2 avec probabilités $1/3, 1/3, 1/3$. Analyser précisément sa complexité. Votre algorithme est-il de type Monte Carlo ou Las Vegas ?

(ii). On suppose qu'on a accès à une source de bits aléatoires indépendants, c'est-à-dire à une suite de variables aléatoires X_i indépendantes et identiquement distribuées, où chaque X_i vaut 0 avec probabilité p et 1 avec probabilité $1 - p$. Malheureusement la valeur de p est inconnue (par exemple, il peut d'agir d'une pièce de monnaie biaisée dont on ignore le biais exact).

Proposer un algorithme pour générer une variable aléatoire Z qui soit égale à 0 avec probabilité exactement $1/2$ et à 1 avec probabilité exactement $1/2$. Analyser précisément sa complexité. Votre algorithme est-il de type Monte Carlo ou Las Vegas ?