

# Algorithmes gloutons

## 1 Le voleur intelligent

Un cambrioleur entre par effraction dans une maison et désire emporter quelques-uns des objets de valeur qui s'y trouvent. Il n'est capable de porter que  $x$  kilos : il lui faudra donc choisir entre les différents objets, suivant leur valeur (il veut bien entendu amasser le plus gros magot possible...).

On suppose dans un premier temps que les objets sont des matières fractionnelles (on peut en prendre n'importe quelle quantité, c'est le cas d'un liquide ou d'une poudre). Il y a  $n$  matières différentes, numérotées de 0 à  $n - 1$ , la  $i$ -ième ayant une valeur  $p.(i)$  par kilo. La quantité disponible de cette matière est  $q.(i)$ . On suppose que toutes les valeurs sont différentes deux à deux.

**Question 1.** *Proposer un algorithme qui renvoie la valeur optimale du butin du voleur. Ce choix est-il unique ? Programmer une fonction voleur : float vect -> float vect -> float -> float qui implémente cet algorithme (on pourra supposer que le tableau  $p$  est trié).*

On suppose maintenant que les objets sont non fractionnables (c'est le cas d'une chaise ou d'un téléviseur). Le  $i$ -ème objet a une valeur  $p.(i)$  et pèse un poids  $q.(i)$ .

**Question 2.** *Proposer une méthode dérivée de la question 1 (sans la programmer). Donne-t-elle un choix optimal ?*

## 2 Réservation SNCF

On suppose que  $n$  personnes veulent prendre le train un jour donné. La personne  $i$  veut prendre le train  $p.(i)$ , où  $p$  est un tableau d'entiers. Les trains sont numérotés de 0 à  $k - 1$ , et partent dans l'ordre de leur numéro. Chaque train peut contenir au plus  $c$  personnes.

**Question 3.** *Écrire une procédure possible : int -> int vect -> int -> bool telle que possible c p k renvoie true si et seulement si, pour tout train  $t \in \{0, \dots, k - 1\}$ , il y a au plus  $c$  personnes qui veulent prendre le train  $t$ .*

On suppose maintenant que la personne  $i$ , si elle ne peut pas prendre le train  $p.(i)$  (parce qu'il est complet), accepte de prendre un des trains suivants.

**Question 4.** *Proposer un algorithme sncf pour affecter chaque personne à un train lorsque cela est possible. Votre procédure renverra un tableau d'entiers donnant, pour chaque client, le numéro du train auquel il a été affecté. L'algorithme devra traiter les clients dans l'ordre : on suppose que les demandes de réservation se font l'une après l'autre, ainsi l'affectation d'un train au client  $i$  ne peut pas dépendre des demandes des clients  $i + 1, i + 2, \dots$  et ne peut être changée une fois qu'elle a été effectuée.*

## 3 Remplissage d'un camion

On dispose de  $n$  marchandises, de poids respectifs  $p.(0), p.(1), \dots, p.(n - 1)$ , où  $p$  est un tableau d'entiers strictement positifs. On supposera ce tableau trié dans l'ordre croissant. On désire remplir un camion dont la charge maximale autorisée est  $x$  avec certaines des marchandises, de manière à obtenir le plus grand poids possible inférieur ou égal à  $x$ . De manière plus abstraite, on cherche en fait la plus

grande valeur inférieure ou égale à  $x$  que l'on puisse obtenir en sommant les éléments d'un sous-ensemble de  $\{p.(0), \dots, p.(n-1)\}$  (on précise bien que c'est la valeur en question que l'on cherche, et non pas le sous-ensemble correspondant). On va concevoir un algorithme qui procède en  $n$  étapes : à l'étape  $i$ , on construira la liste de tous les poids inférieurs ou égaux à  $x$  que l'on peut obtenir en prenant des éléments dans  $p.(0), \dots, p.(i-1)$ . On a besoin pour cela de quelques fonctions préliminaires.

**Question 5.** *Programmez une fonction ajoute : int -> int list -> int list qui prend pour argument un entier a ainsi qu'une liste d'entiers m = [m0 ; ... ; mk-1] et retourne la liste [m0+a ; ... ; mk-1+a].*

**Question 6.** *Écrivez une fonction retire : int -> int list -> int list qui prend l'entier x et une liste m et retourne cette liste dans laquelle les éléments strictement supérieurs à x ont été supprimés.*

**Question 7.** *Définissez enfin une fonction list max : int list -> int qui calcule le maximum d'une liste.*

**Question 8.** *Déduisez-en une fonction camion : int vect -> int -> int qui résout le problème. Que pensez-vous de la complexité de votre algorithme ?*

Pour obtenir un algorithme plus efficace, on relâche nos exigences : on se fixe une valeur  $\epsilon > 0$  donnée, et on cherche maintenant un sous-ensemble de  $\{p.(0), \dots, p.(n-1)\}$  dont la somme  $S$  des éléments est inférieure à  $x$ , et telle que si  $S^*$  est la somme correspondant à la solution optimale, alors  $S \geq S^*(1 - \epsilon)$ .

**Question 9.** *Modifier la procédure camion de sorte qu'à chaque étape  $i$  (consistant à construire la liste  $L_i$  des sommes réalisables à partir de  $\{p.(0), \dots, p.(i-1)\}$  à partir de celle des sommes réalisables à partir de  $\{p.(0), \dots, p.(i-2)\}$ ), la liste  $L_i$  construite par l'algorithme soit :*

- triée

- telle que, si  $s_j$  et  $s_{j+1}$  sont deux entiers consécutifs de  $L_i$ , alors  $s_j \leq s_{j+1}(1 - \epsilon/n)$ .

Pour cela, on introduira une procédure fusion, et on modifiera la procédure retire de manière adéquate.

**Question 10.** *Montrer que la procédure camion ainsi modifiée résout le problème posé. On pourra en particulier montrer par récurrence sur  $i$  que, pour tout  $\sigma$  appartenant à l'ensemble des sommes des parties de  $\{p.(0), \dots, p.(i-1)\}$ , il existe une somme  $\lambda$  appartenant à la liste  $L_i$  construite après l'étape  $i$  telle que  $(1 - \epsilon/n)^i \sigma \leq \lambda \leq \sigma$ .*

**Question 11.** *Borner la complexité du nouvel algorithme en fonction de  $n$ ,  $x$  et  $\epsilon$ .*

## 4 Chaîne maximum d'une permutation

On considère une permutation  $\sigma$  de  $\{0, \dots, n-1\}$ . Soit  $p_i = (i, \sigma(i)) \in \mathbb{Z}^2$ . On dit que  $p_i$  domine  $p_j$  si  $i \geq j$  et  $\sigma(i) \geq \sigma(j)$ .

**Question 12.** *Montrer que la relation de domination est une relation d'ordre partiel.*

On la note  $\geq$ , et on note  $p_i > p_j$  si  $p_i \geq p_j$  et  $p_i \neq p_j$ . Une chaîne est une suite de points  $p_{i_1} > p_{i_2} > \dots > p_{i_k}$ . On définit la hauteur d'un point  $p_i$  comme étant la longueur maximale d'une chaîne dans l'ensemble  $\{p_j$  tels que  $p_i \geq p_j\}$ . Soit  $S_h$  l'ensemble des points de hauteur  $h$ .

**Question 13.** *Montrer que si  $p_i$  est de hauteur  $h > 1$ , alors il existe un point  $p_j, j < i$ , qui appartient à  $S_{h-1}$  et est dominé par  $p_i$ . Montrer qu'alors le point de  $S_{h-1}$  le plus à droite (ie d'abscisse maximum) parmi ceux qui sont à gauche de  $p_i$  (c'est-à-dire d'abscisse strictement inférieure à celle de  $p_i$ ) est aussi dominé par  $p_i$ .*

**Question 14.** *Écrire une procédure hauteurs : int vect -> int vect telle que hauteur  $\sigma$  renvoie le tableau des hauteurs des points définis par la permutation  $\sigma$  (donnée sous la forme d'un tableau). On veillera à ce que la complexité de la procédure ne dépasse pas  $O(nh_{max})$ , où  $h_{max}$  est la hauteur maximale d'un point de l'ensemble défini par  $\sigma$ . Pour cela, on pourra utiliser un tableau auxiliaire adroite contenant, pour chaque hauteur  $h$ , l'indice du point de hauteur  $h$  ayant la plus grande abscisse.*

**Question 15.** *Proposer un algorithme pour trouver une chaîne  $p$  de taille maximale. Quelle est sa complexité ?*