Première NSI - Algorithmique

Les algorithmes gloutons - 2. travaux dirigés

qkzk

2020/08/01

1. Le problème du sac à dos

Un cambrioleur possède un sac à dos d'une contenance maximum de 30 Kg. Au cours d'un de ses cambriolages, il a la possibilité de dérober 4 objets A, B, C et D. Voici un tableau qui résume les caractéristiques de ces objets :

Table 1: premier cambriolage

objet	A	B	C	D
masse	13 kg	12 kg	8 kg	10 kg
valeur marchande	700 €	400 €	300 €	300 €

On ajoute les contraintes suivantes :

- le sac à dos a une contenance de 30 kg
- le cambrioleur cherche à obtenir un gain maximum.

1. Déterminez les objets que le cambrioleur aura intérêt à dérober.
2. Quel critère pourrait-on choisir pour trier les objets ? Proposer un algorithme glouton pour résoudre le problème du sac à dos.
3. Retourne-t-il la solution optimale ?
4. Reprendre le problème avec les objets suivants :

Table 2: second cambriolage

objet	A	B	C	D
masse	35 kg	41 kg	28 kg	39 kg
valeur marchande	70 €	40 €	30 €	30 €

   a. Si le sac peut contenir 100 kg,
   b. Si le sac peut contenir 85 kg.

5. Écrire une fonction Python qui calcule les valeurs massiques d'une liste d'objets passés en paramètre.
6. Écrire une fonction Python qui renvoie le contenu d'un sac à dos depuis une liste d'objets et une contenance de sac à dos passés en paramètres. Cette fonction utilisera l'algorithme glouton présenté plus haut.

2. Le rendu de monnaie.

On considère un jeu de pièce et une somme à rendre. Nous allons étudier deux situations générales, pour lesquelles l'algorithme glouton retourne la solution optimale ou non.

1. On considère un jeu de pièce similaire à l'euro : 1 cts, 2 cts, 5 cts, 10 cts, 20 cts, 50 cts, 1 €, 2 €, 5 €, 10 €, 20 €, 50 €, 100 €, 200 €.

Rappeler l'algorithme glouton qui renvoie les pièces à rendre sous la forme d'une liste à partir d'un jeu de pièces et d'un montant passés en paramètres.

2. Donner les étapes pour le rendu de 71€73
3. Écrire une fonction Python qui traite le problème exposé à la question précédente.

Cette fonction retourne toujours une réponse mais celle-ci n'est pas forcement la meilleure.

3. Proposer un algorithme qui teste toutes les combinaisons possibles et renvoie la meilleure.
4. Comparer les complexités de deux algorithmes : glouton et exhaustif.