PCSI 1 **Devoir surveillé d'informatique n° 2 :** 2023/2024

Jeudi 25 avril

2h

Les fonctions doivent être commentées.

L'indentation doit être représentée par un trait vertical.

Optimisation de rendement d'une entreprise de livraison

Un entreprise de livraison souhaite optimiser le chargement de ses camions pour diminuer ses frais de fonctionnement.

Partie 1 : Optimisation du chargement

Chaque camion de l'entreprise peut charger une cargaison jusqu'à un poids maximal noté P_{max}. L'entreprise dispose de différentes informations provenant de ses clients :

- le poids de chaque produit p_i,
- la valeur v_i associée au transport de chaque produit : c'est-à-dire l'argent gagné par l'entreprise si elle réalise le transport du produit.

En considérant que l'entreprise dispose de n produits, l'entreprise cherche donc à trouver une liste d'indices notée I contenue dans $[\![0, n-1]\!]$ telle que :

$$\sum_{i\in I} p_i \leq P_{max} \text{ (respect du poids maximal)}$$

et

$$\sum_{i\in I} v_i \text{ soit maximal (optimisation du profit pour l'entreprise).}$$

Dans toute la suite, les poids seront donnés en centaine de kilogrammes et les valeurs en centaines d'euros.

On notera P la liste des poids : $P = [p_0, p_1, \ldots, p_{n-1}]$ et V la liste des valeurs : $V = [v_0, v_1, \ldots, v_{n-1}]$.

1. **Un exemple.**
   Dans le cas particulier où $P = [3,2,1,4]$, $V = [4,3,1,9]$ et $P_{max} = 8$, donner la cargaison maximisant le profit de l'entreprise. Que vaut alors le profit ?
2. **Une méthode intuitive pour la résolution du problème.**
   Une méthode intuitive pour tenter d'optimiser le profit de l'entreprise est la suivante : on calcule les ratios $\frac{v_i}{p_i}$, puis on trie les objets par ordre décroissant suivant ces valeurs. Les produits sont alors classés par rentabilité : le premier produit devient le plus rentable "au poids" et ainsi de suite. On ajoute progressivement chaque produit dans la cargaison, dans cet ordre, sans dépasser la limite du poids maximal.
   (a) Définir une fonction `Ratio` ayant pour arguments deux listes P et V, renvoyant la liste des ratios $\frac{v_i}{p_i}$. Calculer la complexité de cette fonction.
   (b) Définir une fonction `TriSelection` ayant pour argument une liste de nombres réels l, renvoyant la liste l triée par ordre croissant en utilisant le principe du tri par sélection. Donner la complexité de cette fonction.
   (c) Définir une fonction `Inverse` ayant pour argument une liste de nombres réels l, renvoyant l'inverse de la liste l. Par exemple, l'inverse de $[1,5,3,4]$ et $[4,3,5,1]$. Cette fonction devra être de complexité linéaire et ne pas avoir d'effet de bord. On justifiera que ces contraintes sont bien vérifiées.
   (d) En utilisant les fonctions précédentes, définir une fonction `TriRatio` ayant pour arguments deux listes P et V, renvoyant la liste des ratios $\frac{v_i}{p_i}$ triés par ordre décroissant.
   Cette fonction ne permettra pas de répondre au problème car elle ne renvoie que la liste des ratios et pas celle des produits.
   (e) En utilisant uniquement la fonction `Ratio`, définir une fonction `TriRatio2` ayant pour arguments deux listes P et V, renvoyant les listes de poids et de valeurs triées par ordre décroissant de la liste des ratios.
   (f) Définir une fonction `Vmax` ayant pour arguments deux listes P et V et une valeur de poids maximal P_{max}, renvoyant la valeur maximale du profit de l'entreprise en suivant la méthode proposée. Calculer la complexité de cette fonction.
   (g) On revient à l'exemple de la question 1 : $P = [3,2,1,4]$, $V = [4,3,1,9]$ et $P_{max} = 8$. Donner le résultat obtenu par la fonction `Vmax`. Commenter le résultat. Quel type d'algorithme a-t-on utilisé ?

3. **Une méthode récursive**

   Nous gardons les notations P, V et P_{max} définies dans la partie précédente. On suppose dorénavant que les poids p_i ainsi que P_{max} sont des entiers.

   Nous introduisons une méthode récursive pour résoudre le problème d'optimisation :

   - pour chacun des produits, deux choix sont possibles, il fait partie de la cargaison ou non ;
   - la récursivité s'effectuera sur le nombre i de produits restants : le premier appel de la fonction se fera en utilisant l'indice n, puis l'indice $n-1$ et ainsi de suite jusqu'à l'indice 0 (correspondant au cas où il n'y a plus de produits) ;
   - pour $i \in [\![0, n]\!]$ et $w \in [\![0, P_{max}]\!]$, on note $S(i, w)$ la valeur maximale cumulée des produits que l'on peut placer dans un camion d'une capacité maximale en poids w avec la liste constituée des i premiers produits.

   On pose alors la relation de récursivité suivante :

$$S(i,w) = \begin{cases} 0 & \text{si } i = 0, \\ S(i-1,w) & \text{si } i > 0 \text{ et } p_{i-1} > w \\ \max(S(i-1,w), v_{i-1} + S(i-1, w-p_{i-1})) & \text{si } i > 0 \text{ et } p_{i-1} \leq w \end{cases}$$

   (a) Justifier les relations précédentes.
   (b) Justifier la terminaison de l'algorithme associé à la relation de récursivité précédente, sachant que la première valeur donnée pour i sera n et la première valeur pour w sera P_{max}.
   (c) Définir une fonction `Max` ayant pour arguments deux réels et renvoyant le maximum parmi ces deux valeurs.
   (d) En utilisant la relation de récursivité, définir une fonction `recur` ayant pour arguments des listes P et V, un indice i et un poids w, et renvoyant $S(i, w)$.
   (e) Comment peut-on utiliser la fonction `recur` pour répondre au problème ?
   (f) Calculer la complexité de la fonction `recur`. Commenter le résultat obtenu.

4. **Amélioration de la méthode récursive**

   On souhaite améliorer la méthode récursive de la partie précédente. Nous allons procéder en mémorisant les calculs déjà effectués. Voici le principe : nous allons stocker les valeurs $S(i, w)$ dans une matrice `Memoire`, de taille $(n+1) \times (P_{max}+1)$, initialisé au départ avec des éléments tous égaux à -1.

   Si la valeur de $S(i, w)$ a déjà été calculée, l'élément d'indice (i, w) de cette matrice `Memoire` ne sera plus égal à -1 : on renverra donc directement sa valeur. Sinon, on le calculera en suivant le principe de la question précédente et on le stockera dans la matrice avant de le renvoyer.

   Nous faisons le choix de représenter les matrices comme des listes de listes.

   (a) Donner l'instruction permettant de créer la matrice `Memoire` initialisé avec des coefficients égaux à -1 en supposant P_{max} et n connus.
   (b) En utilisant ce principe, définir une fonction `recur2` ayant pour arguments des listes P et V, un indice i, un poids w et une matrice `Memoire`, et renvoyant $S(i, w)$. La variable `Memoire` sera considérée comme initialisée avec l'instruction de la question précédente.
   (c) Déterminer une majoration du nombre d'appel récursif de la fonction `recur2`. Commenter le résultat obtenu.

Partie 2 : Identifiant des produits

Dans cette partie, l'opération de calcul de puissances ne sera pas considérée comme élémentaire mais aura un coût correspondant au nombre de produits effectués. Ainsi, l'opération $x**n$ a un coût en $O(n)$.

L'indice i représentant un produit est stocké par l'entreprise en codage binaire sur 8 bits. Par exemple, '00010111' est associé au produit $i = 23$.

1. Donner le codage associé au produit 39.
2. Définir une fonction `identifiant` ayant pour argument le code binaire d'un produit représenté par une chaîne de caractères, renvoyant le numéro du produit. Calculer la complexité de cette fonction.
3. Redéfinir la fonction précédente de façon à ce que sa complexité devienne linéaire.