DM2 - CORRECTION

Une entreprise de livraison dispose de plusieurs locaux en France et chacun possède plusieurs camions de livraisons. Celle-ci souhaite optimiser le chargement de ses camions pour diminuer ses frais de fonctionnement.

1 Partie I - Données liées aux livraisons conservées par l'entreprise

À chaque livraison, l'entreprise stocke des données relatives à celle-ci.

L'entreprise dispose de 20 locaux, numérotés de 1 à 20, disposant chacun d'un certain nombre de camions. Pour faciliter ses livraisons, l'entreprise découpe la France en 30 zones et associe trois zones possibles de livraisons à chaque local.

Ces données sont enregistrées dans une base de données composée de trois tables :

La table livraison constituée des champs suivants :

- date : date de la livraison au format "jj-mm-aaaa" (chaîne de caractères) ;
- heure : heure de la livraison au format : "hh-mm-ss" (chaîne de caractères) ;
- id_client : identifiant du client recevant la livraison (entier) ;
- id_local : identifiant du local de l'entreprise (entier compris entre 1 et 20) ;
- id_commande : identifiant de la commande (chaîne de caractères).

La table client constituée des champs suivants :

- id : identifiant du client (entier) ;
- zone : entier compris entre 1 et 30.

La table local constituée des champs suivants :

- id : identifiant du local (entier compris entre 1 et 20) ;
- zone1 : entier ;
- zone2 : entier ;
- zone3 : entier.

Q1. Donner une clé primaire pour la table livraison. Y-a-t-il d'autres possibilités ?
Il n'y a pas de clé primaire pour la table livraison sauf à en prendre une sur 3 attributs : date, heure et id_client et à considérer qu'il ne peut y avoir deux livraisons le même jour à la même heure chez le même client.

Q2. Écrire une requête SQL permettant d'obtenir les identifiants des clients livrés le 10 janvier 2023.
SELECT id_client FROM livraison WHERE date="10-01-2021"

Q3. Écrire une requête SQL permettant de récupérer les heures de toutes les livraisons ayant eu lieu dans la zone 5 le 2 mars 2023. SELECT date, heure FROM livraison AS liv JOIN client AS c ON liv.id_client=c.id WHERE liv.date="02-03-2021" AND c. zone=5

Q4. Écrire une requête SQL permettant de compter le nombre de livraisons effectuées le 3 février 2023 par des camions dont les locaux ne livrent que dans des zones possibles inférieures ou égales à 10.
SELECT COUNT (*) FROM livraison AS liv J0IN local AS loc ON liv.id_client=loc.id WHERE liv. date="03-02-2021" AND loc.zonel<10 AND loc. zone2<10 AND loc. zone3<10

2 Partie II - 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 fournisseurs :

- le poids de chaque produit p_i (chaque fournisseur propose un seul produit) ;
- 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 de ce produit. En considérant que l'entreprise dispose de n fournisseurs, l'entreprise cherche donc à trouver une liste d'indices notée I contenue dans $\{0, \dots, n-1\}$ telle que :

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

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 centaines de kilogrammes et les valeurs en centaines d'euros.

3 II.1 - Un exemple

Dans cette sous-partie, on suppose que $n = 4$ et que $P_{\max} = 8$ centaines de kilogrammes.

On stocke alors les différentes informations dans trois listes :

— Produits est la liste des produits proposés par les fournisseurs, numérotés de 0 à 3.

Ici Produits $= [0, 1, 2, 3]$;

— Poids est la liste des poids associés : Poids $= [3, 2, 1, 4]$;

— Valeurs est la liste des valeurs associées : Valeurs $= [4, 3, 1, 9]$.

Par exemple, le produit 1 a un poids de deux centaines de kilogrammes et une valeur de trois centaines d'euros.

Les questions Q5., Q6. et Q7. se traitent à l'aide de calculs simples, à faire à la main.

Q5. Expliquer pourquoi une cargaison constituée d'un, de deux ou de quatre produits ne répond pas au problème posé, c'est-à-dire ne maximise pas le profit fait par l'entreprise en respectant la condition donnée sur le poids maximal.
Lorsqu'on ne met que 1 produit, on a une valeur maximale de 9.

Lorsqu'on ne met que 2 produits, on a une valeur maximale de 13 (produits 1 et 4).
Or il est clair qu'en mettant les produits 1,3 et 4 on a une valeur plus élevée : 14.

Lorsqu'on met 4 produits, le poids maximal est dépassé.

Q6. Donner toutes les cargaisons de trois produits respectant le poids maximal. On donnera à chaque fois le profit fait par l'entreprise.
Les cargaisons de 3 produits :
Produits 1,2 et 3 V = 8
Produits 1,3 et 4 V = 14
Produits 2,3 et 4 V = 13

Q7. Quelle est la cargaison maximisant le profit de l'entreprise ? Que vaut le profit dans ce cas ?
On remarque que la cargaison maximisant le profit est 1,3,4 avec une valeur V=14

4 II.2 - Une approche exhaustive

Soit $n \in \mathbb{N}^*$. On garde les notations de la sous-partie précédente dans le cas général :

— Produits $= [0, 1, 2, \ldots, n-1]$ est la liste des produits
— Poids $= [p_0, \ldots, p_{n-1}]$ est la liste des poids associés aux produits ;
— Valeurs $= [v_0, \ldots, v_{n-1}]$ est la liste des valeurs associées aux produits.

Q8. Définir une fonction listeProduits ayant pour argument un entier naturel non nul n créant et renvoyant la liste Produits.

```
def ListeProduits(n):
    return [i for i in range(1,n+1)]
```

Une combinaison de i éléments de $\{0, \ldots, n-1\}$ est un sous-ensemble de $\{0, \ldots, n-1\}$ contenant i éléments.

En Python, une combinaison sera représentée par une liste ordonnée dans l'ordre croissant : par exemple la combinaison $\{1, 4, 5\}$ sera représentée par la liste $[1, 4, 5]$.

Une approche exhaustive pour optimiser le chargement consiste à tester toutes les combinaisons de i produits parmi n et à garder celle qui optimise le rendement : les deux contraintes (1) et (2) doivent donc être respectées.

Q9. Écrire une fonction Python teste_cargaison qui reçoit en arguments les listes Poids, Valeurs, Pmax, une combinaison C de la liste Produits et renvoie :

- Une valeur numérique correspondant à la valeur de la cargaison si la combinaison C de produits est compatible avec la contrainte (1).
- La valeur -1 si la combinaison proposée est incompatible avec la contrainte (1).

```
def teste_cargaison(Poids, Valeurs, Pmax,C):
    V=0
    P=0
    for p in C:
        V=V+Valeurs[p]
        P=P+Poids[p]
    if P<=Pmax:
        return(V)
    return(-1)
```

On admet que l'on dispose d'une fonction Python combinaisons, d'argument une liste L, qui renvoie la liste de toutes les combinaisons d'éléments de la liste L.

Par exemple :

```
>>> L = [0,1,2]
>>> LC = combinaisons(L)
>>> LC
[[],[0],[1],[2], [0, 1],[0,2], [1,2],[0,1,2]]
```

Q10. Voici le code partiel de la fonction optimise d'arguments Produits, Poids, Valeurs et Pmax, qui renvoie la combinaison assurant le meilleur remplissage ainsi que la valeur de la cargaison associée.

Compléter le code de cette fonction optimise en fonction des indications précédentes.

```
def optimise(Produits ,Poids ,Valeurs ,Pmax):
    Csol =[] # combinaison solution
    Vsol=0 # valeur associee de la cargaison
    LC =combinaisons(Produits)
    for C in LC:
        val=teste_cargaison(Poids ,Valeurs ,Pmax ,C)
        if val !=-1 and val>Vsol:
            Vsol=val
            Csol=C
    return Csol ,Vsol
```

Q11. On admet qu'il y a 2^n combinaisons d'éléments de la liste Produits (y compris la combinaison vide de 0 élément parmi n).

Si $n = 50$ et que tester la validité d'une combinaison prend 10^{-9} secondes, combien de temps faudra t-il approximativement pour optimiser le remplissage du camion ?

On prendra $2^{50} \approx 10^{15}$ et 1 heure $\approx 4 \times 10^3$ secondes.

Il y a 2^{50} itérations de la boucle for.
A chaque itération est appelée la fonction teste-cargaison dont le délais d'exécution est de 10^{-9} secondes.
$2^{50} \times 10^{-9} \approx 10^{15} \times 10^{-9} = 10^6$ secondes.
Il faut donc $\frac{10^6}{4\times10^3} = 250$ heures pour optimiser le remplissage du camion.

La démarche précédente est donc inefficace pour un nombre relativement élevé d'objets.

Aussi il est préférable de mettre en place une méthode plus rapide dont on étudiera ensuite l'optimalité.

5 II.3 - Une méthode intuitive pour la résolution du problème

On garde les notations de la sous-partie précédente.

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.

Q12. Écrire une fonction ratio ayant pour arguments deux listes Poids et Valeurs où Poids correspond à la liste des poids et Valeurs correspond à la liste des valeurs, renvoyant la liste des ratios $\frac{v_i}{p_i}$.

```
def Ratio(Poids,Valeurs):
    rapports=[]
    for i in range(len(P)):
         rapports.append(Valeurs[i]/Poidsids[i])
    return rapports
```