teaching sciences for innovation

# Proposition de corrigé

Concours : Concours Commun INP

Année : 2021

Filière : TSI

Épreuve : Informatique

Ceci est une proposition de corrigé des concours de CPGE, réalisée bénévolement par des enseignants de Sciences Industrielles de l'Ingénieur et d'Informatique, membres de l'UPSTI (Union des Professeurs de Sciences et Techniques Industrielles), et publiée sur le site de l'association :

https://www.upsti.fr/espace-etudiants/annales-de-concours

## A l'attention des étudiants

Ce document vous apportera des éléments de corrections pour le sujet traité, mais n'est ni un corrigé officiel du concours, ni un corrigé détaillé ou exhaustif de l'épreuve en question.

L'UPSTI ne répondra pas directement aux questions que peuvent soulever ces corrigés : nous vous invitons à vous rapprocher de vos enseignants si vous souhaitez des compléments d'information, et à vous adresser à eux pour nous faire remonter vos éventuelles remarques.

## Licence et Copyright



En cas de doute, n'hésitez pas à nous contacter à : corrigesconcours@upsti.fr.

## Informez-vous !

Retrouvez plus d'information sur les Sciences de l'Ingénieur, l'orientation, les Grandes Ecoles ainsi que sur les Olympiades de Sciences de l'Ingénieur et sur les Sciences de l'Ingénieur au Féminin sur notre site : www.upsti.fr

L'équipe UPSTI

# Optimisation de rendement d'une entreprise de livraison

Corrigé UPSTI

## I - Optimisation du chargement

### I.1 - Un exemple

**Question 1** 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.

Le plus lourd des produits ne remplit pas le camion, on pourrait en mettre un deuxième ce qui augmenterait le profit.

Avec les 2 produits les plus lourds 400kg et 300kg, on pourrait encore ajouter un produit à 100kg pour maximiser le profit.

Enfin avec 4 produits (masse total de 1000 kg), on dépasse le poids maximal de 800kg.

**Question 2** Donner toutes les cargaisons de trois produits respectant le poids maximal. On donnera à chaque fois le profit fait par l'entreprise.

[1,2,3] donne 3+2+1=6*100kg de charge < 800kg et un profit de 4*3*1=8*100€.

[1,3,4] donne 3+1+4=8*100kg (poids maximal atteint) et un profit de 4+1+9=14*100€.

[2,3,4] donne 2+1+4=7*100kg <800kg et un profit de 3+1+9=13*100€.

**Question 3** Quelle est la cargaison maximisant le profit de l'entreprise? Que vaut le profit dans ce cas ?

La cargaison la plus profitable est [1,3,4] avec un profit de 1400€.

### I.2 - Une méthode intuitive pour la résolution du problème

**Question 4** Définir une fonction ListeProduits ayant pour argument un entier naturel non nul $n$ et renvoyant la liste Pr.

```
def ListeProduits (n):
    Pr=[0]*n
    for i in range (n):
        Pr[i] = i+ 1
    return Pr
```

**Question 5** Définir une fonction Ratio ayant pour arguments deux listes P, V où P correspond à la liste des poids et V correspond à la liste des valeurs, renvoyant la liste des ratios $\frac{v_i}{p_i}$.

```
def Ratio(P,V):
    n = len (P)        # on suppose que len(P)==len(V)
    ratios = [0]*n
    for i in range (n):
        ratios[i] = V[i]/P[i ]
    return ratios
```

**Question 6** On exécute Tri (L) avec L= [3, 5, 2, 1]. Combien y a t-il d'itérations de la boucle for? Donner la valeur de L à la fin de chaque itération de la boucle for.

n- 1 itérations (3 pour l'exemple) où n est le nombre d'éléments de L.

fin de 1ère itération : [3, 5, 2, 1]

fin de 2ème itération : [2, 3, 5, 1]

fin de 3ème itération : [1, 2, 3, 5]

**Question 7** Ce tri fonctionnerait-t-il pour une chaîne de caractères dont chaque caractère est un entier? Justifier.

Ce tri ne fonctionnerait pas avec des chaines de caractères car les chaines de caractères ne sont pas assignables par leur indice : **L[j]=x** génère par exemple une erreur si L est une chaine de caractères.

**Question 8** Quelle est la méthode de tri utilisée dans la fonction Tri ? Donner sa complexité (en nombre de comparaisons) dans le pire des cas et dans le meilleur des cas. *On justifiera soigneusement les complexités données.*

La méthode de tri proposé est le tri par insertion.

Dans le pire des cas sa complexité est quadratique en $o(n^2)$ (L triée en sens inverse) :

- 1ère itération x est plus petit que L[0] donc on parcourt le while après ce couple de comparaisons. Le dernier test est faux car j est nul ce qui ajoute un couple de comparaisons supplémentaire. (soit finalement 4 comparaisons) : u(1)=4
- ie itération : x est plus petit que les i termes à comparer soit i couples de comparaisons auxquelles on ajoute le test de fin de boucle lié à la nullité de j : u(i)=2i+2 (nombre de comparaisons de l'algorithme même si en pratique pour ce ET la deuxième comparaison n'est pas évaluée lorsque la première est fausse).
- la somme de cette suite arithmétique vaut :

$$C(n) = \left(\frac{u(n-1)+u(1)}{2}\right)(n-1) termes = \frac{2(n-1)+2+4}{2} * (n-1) = (n+2)(n-1) \sim o(n^2).$$

Dans le meilleur des cas, la complexité du tri est linéaire en o(n) (L triée) :

- 1ère itération x : est plus grand que L[0] donc on sort du while à la 1ère itérations :u(1)=2 comparaisons
- ième itération : x est plus grand le terme qui le précède donc : u(i)= 2 comparaisons
- Finalement C(n)=2*(n-1) $\sim o(n)$.

**Question 9** Définir une fonction Inverse ayant pour argument une liste de nombres réels L et renvoyant l'inverse de celle-ci. Par exemple, l'inverse de [1, 5, 3, 4]est [ 4, 3, 5, 1].

L'utilisation de L [: : -1] n'est pas autorisée.

```
def Inverse(L) :
    n = len (L)
    Ls=[0]*len(L)
    for i in range (n):
        Ls[i]=L[n-1-i]
    return Ls
```

**Question 10** On souhaite trier une liste de poids P et une liste de valeurs V associées à une liste de produits en suivant l'ordre décroissant de la liste des ratios $\frac{v_i}{p_i}$. Justifier que les fonctions Ratio, Tri et Inverse ne permettent pas de répondre simplement au problème posé.

Le tri n'est pas si simple car dans l'ordonnancement, il faut prendre aussi en compte la capacité de charge Pmax du camion qui va modifier l'optimisation du profit. Cette optimisation ne se fera pas uniquement en chargeant les produits les plus rentables mais en complétant avec des produits moins rentables mais moins lourds par exemple.

A la lecture des questions suivantes, on peut aussi comprendre que les différentes listes doivent être triées en même temps (ce qui n'était pas le cas de la fonction Tri).

**Question 11** Écrire, à l'aide des fonctions Ratio et Inverse, une fonction Tri2 ayant pour arguments une liste de poids P et une liste de valeurs V associées à une liste de produits. Cette fonction renverra les listes de poids et de valeurs triées par ordre décroissant de la liste des ratios.

```
def Tri2 (P,V) :
    ratios = Ratio(P,V)
    for i in range(1,len(ratios)):
        x=ratios[i]
        y=P[i],V[i]
        j=i
        while j>0 and x<ratios[j-1]:
            ratios[j]=ratios[j-1]
            P[j],V[j]= P[j-1],V[j-1]      # permutations identiques à celle des ratios
            j = j-1
        ratios[j]=x
        P[j],V[j]=y
    return Inverse(P),Inverse(V)   # inversion pour que les listes soit triées par ordre décroissant des ratios
```

**Question 12** Compléter la définition de la fonction Vmax ayant pour arguments les listes de poids P et de valeurs V et le poids maximal $P_{max}$ du chargement et renvoyant la valeur maximale du profit de l'entreprise en suivant la méthode proposée.

```
def Vmax(P,V,Pmax) :
    P2,V2=Tri2(P,V)
    SP=0
    SV=0
    i=0
    while SP+P2[i]<Pmax and i<len(P)-1:
        SP=SP+P2[i]
        SV=SV+V2[i]
        i=i+1
    return SV
```

**Question 13** On souhaite appliquer cette méthode en utilisant les listes de poids et de valeurs de la sous-partie 1.1. Donner la liste des ratios, les listes de poids et de valeurs obtenues à l'aide de la fonction Tri 2 ainsi que le profit obtenu. Commenter le résultat.

ratios=[1.33 , 1.5, 1, 2.25]

Pr = [4,2,1, 3]

P2 =[4 ,2, 3, 1]

V2=[9, 3, 4, 1]

SP= 4+2=6 soit 6 *100kg

SV = 12 soit 12**100€

Commentaires : il manque le produit à poids réduit (malgré son faible ratio) pour compléter la cargaison.

## I.3 - Une méthode récursive

On pose alors la relation de récursivité suivante :

$$S(i,\omega) = \begin{cases} 0 & \text{si } i = 0 \\ S(i-1,\omega) & \text{si } i > 0 \text{ et } p_i > \omega \\ \max(S(i-1,\omega), v_i + S(i-1,\omega-p_i)) & \text{si } i > 0 \text{ et } p_i < \omega. \end{cases} \quad (3)$$

**Question 14** Justifier les relations précédentes dans les trois cas.

Justification pour i=0 : Il n'y a plus de produit à ajouter, la valeur à embarquer est nulle

Justification pour i>0 et $p_i$>ω : on ne peut pas embarquer un produit de poids supérieur au poids total autorisé donc on cherche un produit de poids inférieur.

Justification pour i>0 er $p_i$<ω : on conserve l'option la plus profitable entre :

- la valeur de tous les produits suivants dans la limite du poids w mais sans le poids $p_i$
- la valeur du produit $p_i$ additionné aux valeurs admissibles dans la charge encore disponible (w –$p_i$)

Remarque : on devine que S représente la valeur de la cargaison par homogénéité de la dernière proposition (3) de S avec vi.

**Question 15** 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 Pmax.

A chaque appel récursif, l'indice d'appel est décrémenté (2e et 3e proposition) et donc va terminer à 0, ce qui mettra fin aux appels récursifs (1ère proposition).

**Question 16** Définir une fonction Max ayant pour arguments deux réels et renvoyant le maximum parmi ces deux valeurs. Il est interdit d'utiliser la fonction max prédéfinie dans Python.

```
def Max(a,b):
    if b>a:
        s=b
    else:
        s=a
    return s
```

**Question 17** En vous basant sur la relation (3 ), compléter la définition de la fonction récursive recur ayant pour arguments les listes de poids et de valeurs P et V, un indice *i*, un poids *w*, et renvoyant *S(i,w)*.

```
def recur(P,V,i,w) :
    if i==0:
        return 0
    if P[i-1]>w:
        return recur(P,V,i-1,w)
     else :
        return Max ( recur(P,V,i-1,w) ,  V[i-1]+recur(P,V,i-1,w-P[i-1] ) )
```

Remarque : la difficulté ici réside dans le fait que P et V ont n valeurs alors que la fonction recur sera exécutée n+1 fois pour atteindre 0 (décalage des indices par rapport à la relation (3) du sujet).

**Question 18** Donner une série d'instructions utilisant la fonction recur et permettant de déterminer le profit de la sous-partie 1.1.

```
VS=recur(P,V,Pr[-1],Pmax)     # donne Vs= 14 (la valeur maximale effectivement possible).
```

## I.4 - Amélioration de la méthode récursive

**Question 19** Donner l'instruction permettant de créer le tableau Memoire initialisé avec des coefficients égaux à -1, en supposant Pmax et n connus.

```
Memoire =  [[-1]*(Pmax+1)] * (n+1)          # listes de (n+1) lignes de (Pmax+1) colonnes de chiffres-1
```

**Question 20** Compléter la fonction recur2 en suivant le principe expliqué et permettant d'améliorer la fonction recur. La variable Memoire sera utilisée comme une variable globale.

```
def recur2(P,V,i,w,Memoire) :
    if i==0:
        return 0
    if Memoire [i] [w] >-1:
        return Memoire [i] [w]
```

```
    if P[i-1]>w:
        Memoire[i] [w]=recur2(P,V,i-1,w,Memoire)
        return Memoire[i] [w]
    else:
        if Memoire[i-1] [w]==-1:
            Memoire [i-1] [w] == recur2(P,V,i-1,w,Memoire)
        if Memoire[i-1] [w-P[i-1]]==-1:
            Memoire[i-1] [w-P[i-1]]=recur2(P,V,i-1,w-P[i-1],Memoire)
    a=max(Memoire[i-1] [w] ,V[i-1]+Memoire[i-1] [w-P[i-1]])
    Memoire[i] [w]=a
    return a
```

Fonction que l'on utilise par l'instruction suivante : recur2(P,V,Pr[-1],Pmax,Memoire)

Remarque : les 3 dernières lignes devraient encore être indentées dans le **else** mais comme les **if** initiaux mettent fin à la fonction avec leurs **return**, cela n'empêche pas la fonction de donner le bon résultat (ici 14 encore).

## I.5 - Données liées aux livraisons conservées par l'entreprise

**Question 21** Donner une clé primaire pour la table **livraison.**

Aucun des attributs existant ne peut convenir pour une clé primaire de la table livraison qui doit être unique pour chaque livraison (il est probable que plusieurs livraisons aient lieu à la même seconde pour des jours différents, il y aura plusieurs livraisons par jour, plusieurs clients et plusieurs locaux).

Une clé primaire peut être constituée à partir des attributs {date, heure, id_client} si l'on considère qu'un client ne peut pas être livré 2 fois à la même seconde le même jour. Par analogie aux autres tables, on pourrait aussi ajouter un attribut entier nommé id.

**Question 22** Écrire une requête SQL permettant d'obtenir les identifiants des clients livrés le 10 janvier 2021.

```
SELECT id_client FROM livraison WHERE date=="10-01-2021";
```

**Question 23** Écrire une requête SQL permettant de récupérer les dates et les heures de toutes les livraisons ayant eu lieu dans la zone 5 le 2 mars 2021.

```
SELECT date, heure
FROM livraison JOIN client ON id_client==client.id
WHERE zone == 5 AND date =="02-03-2021";
```

Remarque : le préfixe client de client.id est précisé car plusieurs tables possèdent ce nom d'attribut.

**Question 24** Écrire une requête SQL permettant de compter le nombre de livraisons effectuées le 3 février 2021 par des camions dont les locaux ne livrent que dans des zones possibles inférieures ou égales à dix.

```
SELECT COUNT(*) FROM livraison JOIN local ON id_local==local.id

WHERE ( zone1<= 10 AND zone2<= 10 AND zone3<= 10) AND date =="03-02-2021";
```

**Question 25** Donner le codage associé au client 39.

| Poids binaire | 128= $2^7$ | 64 | 32 | 16 | 8 | 4 | 2 | 1= $2^0$ |
|---|---|---|---|---|---|---|---|---|
| Nombre binaire (un octet) | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |

```
def Identifiant(Bin):
    '''Bin est est une chaine de caractères
     constituée de 0 et 1'''
    S=0
    for i in range(len(Bin)):
        S=S+Bin[i]*2**(len(Bin)-i)
    return S
```

**Question 26** Trouver les deux erreurs dans le code de la fonction précédente.

Bin[i] ne peut pas être utilisé en l'état dans une opération arithmétique car c'est une chaine de caractère. Il faut convertir cette chaine en entier par l'instruction **int(Bin[i])**.

Il y a un décalage dans les indices de la puissance de 2 : le premier indice doit être 7 et non 8 soit l'exposant **len(Bin)-i-1**

**Question 27** Écrire une fonction Identifiant2, qui donne le même résultat que la fonction Identifiant avec une meilleure complexité. Le rombre de multiplications devra être linéaire suivant la longueur de Bin.

```
def identifiant2(Bin):
    S= 0
    for i in range(len(Bin)):
        S=S*2+int(Bin[i])
    return S
```

**Question 28** Si l'entreprise souhaite aussi stocker la zone de chaque client en codage binaire, donner le nombre de bits minimal nécessaire.

Le nombre de zones est 30. On peut coder ce nombre sur 5 bits puisque qu'on peut alors coder $32 - 1 = 31$ valeurs différentes.

FIN