CCINP TSI Informatique 2021 : Correction

Q1.

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.

Q2.

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

Q3.

On remarque que la cargaison maximisant le profit est 1,3,4 avec une valeur V=14

Q4.

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

Q5.

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

Q6.

Il y a 3 boucles **for** de 1 à `len(L)-1=4-1=3`

À la fin de la 1ère boucle `L=[3,5,2,1]`

À la fin de la 2ème boucle `L=[2,3,5,1]`

À la fin de la 3ème boucle `L=[1,2,3,5]`

Q7.

On ne peut pas écrire dans une chaîne de caractère (non mutable) donc les affectations `L[j]=` ne fonctionneraient pas

Q8.

C'est un tri par insertion.

Complexité dans le pire des cas : tableau trié dans l'ordre inverse :

On réalise $n-1$ boucles **for** et chaque boucle **while** est réalisée i fois soit un nombre de boucles $\sum_{i=1}^{n-1} i=\frac{n(n-1)}{2}$. Le nombre d'instructions est proportionnel au nombre de boucles et la complexité est donc $\Theta(N^2)$.

Complexité dans le meilleur des cas : tableau déjà trié :

On réalise $n-1$ boucles **for** et pas de **while** la complexité est donc $\Theta(N)$.

Q9.

```
def Inverse(L):
    N=len(L)
    renverse=[]
    for i in range(N):
        renverse.append(L[N-1-i])
    return renverse .py
```

Q10.

`Tri` trie la liste suivant ses valeurs, on pourra donc trier les ratios mais pas simultanément `P` et `V` en parallèle.

Q11.

Pourquoi ne pas adapter l'algo de tri par insertion en décroissant directement au lieu d'utiliser inverse ?

```
def Tri2(P,V):
    rapports=Ratio(P,V)
    for i in range(len(P)):
        x=rapports[i]
        Pval=P[i]
        Vval=V[i]
        j=i
        while j>0 and x<rapports[j-1]:
            rapports[j]=rapports[j-1]
            P[j]=P[j-1]
            V[j]=V[j-1]
            j=j-1
        rapports[j]=x
        P[j]=Pval
        V[j]=Vval
    return Inverse(P),Inverse(V)
```

Q12.

```
def Vmax(P,V,Pmax):
    P2,V2=Tri2(P,V)
    SP=0
    SV=0
    i=0
    while i<len(P) and SP+P2[i]<=Pmax: #tant qu'on a des produits
        SP=SP+P2[i]                    #et que le poids ajouté ne fait
        SV=SV+V2[i]                    #pas dépasser Pmax
        i=i+1
    return SV
```

Q13.

`Tri2` nous permet d'obtenir les ratios : `[2.25,1.5,1.3,1]` avec les poids `[4,2,3,1]` et les valeurs `[9,3,4,1]`

La fonction `Vmax` renvoie alors `12` avec un chargement de 2 produits, l'algorithme choisi ne donne pas la solution optimale.

Q14.

$i=0$, il n'y a pas de produits, donc la valeur est nulle.

$i>0$ et $p_i>\omega$ Si on charge le produit le poids sera dépassé donc le produit ne peut être pris la valeur est celle obtenue avec les $i-1$ autres produits.

$i>0$ et $p_i<\omega$ le produit peut être pris car le poids ne sera pas dépassé.

Suivant que l'opération augmente la valeur totale ou pas :

Soit on prend le produit alors la valeur est celle du produit plus la valeur du chargement des $i-1$ autres produits à répartir avec un poids restant diminué du poids du produit i ($v_i+S(i-1,\omega-p_i)$)

Soit on ne prend pas le produit et la valeur est celle des $i-1$ autres produit à répartir avec le même poids restant que précédemment puisqu'on n'a pas « consommé » de poids avec le produit i ($S(i-1,\omega)$)

Q15.

L'algorithme se termine lorsque $i=0$ il faut donc s'assurer qu'on atteindra 0 en partant de n , ce qui est le cas car lorsque $i>0$ on fait appelle à la fonction avec $i-1$ à chaque fois.

Q16.

```
def Max(a,b):
    if a<b:
        return b
    else:
        return a
```

Q17.

```
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]))
```

Q18.

```
Pex=[3,2,1,4]
Vex=[4,3,1,9]
print(recur(Pex,Vex,4,8))
```

Q19.

L'énoncé demande UNE instruction ce n'est pas du tout facile pour un étudiant de TSI lambda.

```
Mem=[[-1 for i in range(Pmax+1)] for j in range(n+1)]
```

Q20.

Rmq : Memoire n'est pas une variable globale, il y a un problème d'indentation des 3 dernières lignes dans le document réponse, le document réponse propose max au lieu d'utiliser logiquement Max de la Q16.

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

Q21.

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.

Q22.

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

Q23.

```
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
```

Q24.

```
SELECT COUNT(*) FROM livraison AS liv JOIN local AS loc ON
liv.id_client=loc.id WHERE liv.date="03-02-2021" AND loc.zone1<10 AND
loc.zone2<10 AND loc.zone3<10
```

Q25.

39 s'écrit en binaire 00100111

Q26.

Ligne 6 : il faut écrire `int(Bin[i])` pour convertir le caractère en entier.

Ligne 6 : il faut écrire `len(Bin)-i-1` dans la puissance de 2 pour convertir correctement.

Q27.

```
def Identifiant2(Bin):
    S=0
    deuxpuissance=1 #2**0
    for i in range(len(Bin)):
        S=S+int(Bin[len(Bin)-1-i])*deuxpuissance
        deuxpuissance=deuxpuissance*2
    return S
```

Il y a deux multiplications par boucle **`for`**, soit `2*len(Bin)` multiplications qui est bien linéaire suivant la longueur de `Bin`.

Q28.

zone est un entier entre 1 et 30 il suffit donc de 5 bits (valeur maximale : $2^5-1=31$) pour le stocker en binaire.