

## TP 3

### SUJET CCINP TSI 2021 - SUR LE PROBLÈME DU SAC À DOS

```
## Question 1
''' Le produit avec le plus de valeur est le 4 et l'on peut lui ajouter un autre produit.

Les deux rapportant le plus sont 1 et 4 mais on peut encore ajouter le 3.

Donc on ne maximise pas le profit avec 1 ou 2 objets.

Si l'on prend les 4 objets alors on dépasse le poids maximal. '''

## Question 2
''' Avec 1, 2 et 3: on a un profit de 8.
    Avec 1, 2 et 4: on dépasse le poids autorisé.
    Avec 1, 3 et 4: on a un profit de 14.
    Avec 2, 3 et 4: on a un profit de 13. '''

## Question 3
''' Le choix maximal est constitué des objets 1, 3 et 4 (poids de 8) et le profit est 14. '''

## Question 4

def ListeProduit(n):
    return [k for k in range(1, n+1)]

def ListeProduit_bis(n):
    return list(range(1, n+1)) # moins l'esprit du programme

def ListeProduit_ter(n):
    L = []
    for k in range(1, n+1):
        L.append(k)
    return L

## Question 5

def Ratio(P, V):
    return [V[k]/P[k] for k in range(len(P))]

def Ratio_bis(P, V):
    L = []
    for k in range(len(P)):
        L.append(V[k]/P[k])
    return L
```

```

## Question 6

''' Puisque len(L) = 4, il y a 3 itérations de la boucle for.
Après le cas i = 1: L = [3, 5, 2, 1].
Après le cas i = 2: L = [2, 3, 5, 1].
Après le cas i = 3: L = [1, 2, 3, 5]. '''

## Question 7

''' Cela ne fonctionne pas avec des chaînes de caractères
car aux lignes 7 et 9 il y a des modifications qui ne sont pas
possibles sur une chaîne de caractères (type "non mutable"). '''

## Question 8

''' Il s'agit du tri par insertion.

Notons n la longueur de L.

Il y a n-1 itérations de la boucle for.
À chaque itération:
- il y a 3 affectations
- et une boucle while avec un test j>0 et une comparaison x < L[j-1].

On compte le nombre de comparaisons du type x < L[j-1].

Dans le meilleur des cas, la liste est déjà triée donc la condition
x<L[i-1] est toujours fausse et il n'y a qu'une seule comparaison
par itération de la boucle for.

Donc dans le meilleur des cas, complexité : O(n).

Dans le pire des cas, j diminue systématiquement jusqu'à 0
c'est le cas d'une liste totalement en sens inverse.
Il y a alors 1+2+...+(n-1) comparaisons i.e. (n-1)*n/2.

Dans dans le pire des cas, complexité : O(n**2).
'''

## Question 9

def Inverse(L):
    M = []
    n = len(L) - 1
    while n >= 0:
        M.append(L[n])
        n = n - 1
    return M

def Inverse_bis(L):
    M = []
    for k in range(len(L)):
        M.append(L[len(L)-k-1])
    return M

def Inverse_ter(L):
    return [L[len(L)-k-1] for k in range(len(L))]

def Inverse_quarto(L): # technique inspirée des piles
    Lc = [x for x in L]
    LI = []
    while Lc != []:
        LI.append(Lc.pop())
    return LI

```

```

## Question 10
''' La fonction Tri ne trie qu'une seule liste
    or on veut trier P et V en fonction du tri de V/P
    donc il faut réécrire la fonction de tri en triant
    en même temps toutes les listes. '''

## Question 11

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

## Question 12

# version correspondant au document réponse

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

# autre version plus conforme à l'algorithme

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

## Question 13
''' Les ratios valent: 4/3, 3/2, 1 et 9/4.
    La fonction Tri2 donne pour les ratios [2.25, 1.5, 1.33, 1],
        pour les poids [4, 2, 3, 1] et pour les valeurs [9, 3, 4, 1].
    La fonction Vmax renvoie alors 12 (car 4 + 2 ? 8 mais 4 + 2 + 3 > 8).
    Cette solution n'est donc pas optimale.
'''

```