TP 22- Proposition de solutions

Solution 1 Rendu de monnaie

Entrée : $x \in \mathbb{N}$ et P une liste décroissante d'entiers

$L \leftarrow$ liste vide
$k \leftarrow$ position du premier élément de P
Tant que $x > 0$ faire
  Si $x < P[k]$ alors
    $k \leftarrow k+1$
  Sinon
    ajouter $P[k]$ à la liste L
    $x \leftarrow x - P[k]$
  FinSi
FinTantQue

Sortie : L

```
def monnaie(x,P=[500,200,100,50,20,10,5,2,1]):
    k,L=0,[]
    while x>0:
        if P[k] > x:
            k=k+1
        else:
            x=x-P[k]
            L. append(P[k])
    return L
```

```
def monnaie_R(x,P=[500,200,100,50,20,10,5,2,1]):
    if x==0:
        return []
    if x<P[0]:
        return Monnaie(x,P[1:])
    else:
        return [P[0]]+Monnaie(x-P[0],P)
```

Solution 2 Considérons la décomposition de 48 :
- avec la méthode gloutonne : 30+12+6 (3 pièces)
- décomposition optimale : 24+24 (2 pièces)

Solution 3 Sélection d'activités

```
def activites(L):
    T=sorted(L,key=lambda t:t[2])
    A,k=[T[0]],1
    while k<len(T):
        if T[k][1]>=A[-1][2]:
            A.append(T[k])
        k=k+1
    return A
```

Solution 4 Allocation de salles

Considérons une approche que reprend le procédé de choix d'activités, en ouvrant une nouvelle salle de cours à chaque parcours jusqu'à avoir affecté chaque cours.

```
def allocation(L):
    T=sorted(L,key=lambda t:t[2])
    C=[]
    while len(T)>0:
        A,k=[T[0]],1
        T.pop(0)
        while k<len(T):
            if T[k][1]>=A[-1][2]:
                A.append(T[k])
                T.pop(k)
            else:
                k=k+1
        C.append(A)
    return C
```

Une version travaillant avec une progression parallèle sur les classes ouvertes ou à ouvrir :

```
def allocation2(L):
    T=sorted(L,key=lambda t:t[2])
    C=[[T[0]]]
    for k in range(1,len(T)):
        cond=True
        for j in range(len(C)):
            if T[k][1]>=C[j][-1][2]:
                C[j].append(T[k])
                cond=False
                break
        if cond:
            C.append([T[k]])
    return C
```

Solution 5 Problème du sac à dos

Il s'agit de reprendre la méthode du rendu de monnaie sur une critère à définir :

1. le nombre d'objet : on remplit selon les poids croissants
2. l'importance des objet : on rempli selon les valeurs décroissantes
3. on introduit une notion de *valeur massique* en ordonnant les objets selon la quantité $\frac{\text{valeur}}{\text{masse}}$. Il convient premièrement de considérer un paramètre d'entrée `M` désignant le poids maximal du sac à dos en négligeant la problématique du volume des objets. Les triplets `[Nom de l'objet,valeur, poids]` sont transformés en 4-uplets en ajoutant la valeur massique. Ensuite il suffit de trier par valeur massique décroissante et d'appliquer l'algorithme glouton.