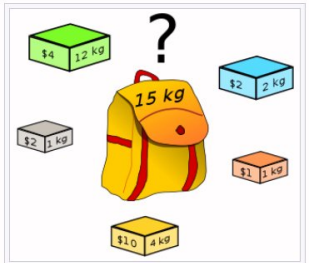
TD SAC A DOS
Algorithmes Gloutons

Problématique générale : Ti' Payet doit traverser l'océan indien avec un bateau et il a droit à une charge maximale de 31181 kg. Il doit transporter un maximum de containers dont la masse et la valeur sont variables. Il compte revendre au meilleur prix ses marchandises.

Comment optimiser le remplissage du bateau afin de ne pas dépasser la capacité totale de 31181 kg tout en ayant la plus grande valeur possible ?



Le problème du sac à dos : quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 15 kg autorisés ?

Ti'Payet se rend compte que le problème est similaire à un problème étudié en classe d'informatique quand il était plus jeune : C'est le problème du sac à dos : quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 15 kg autorisés ? Mais cela fait bien longtemps qu'il n'a plus codé et il a besoin de ton aide...

Il a retrouvé dans ses vieux cours plusieurs programmes, il aimerait bien les tester pour trouver le meilleur compromis, mais les termites ont mangé des morceaux de feuilles. Sauras-tu aider ti'Payet à reconstituer les programmes à partir des éléments dont tu disposes ?

Ti'Payet propose de réaliser un premier algorithme :

1- Algo1 : Classement suivant les masses

Algorithme à coder :

a - On classe les objets dans une liste de la masse la plus faible à la masse la plus élevée

b - On parcours la liste en additionnant les valeurs et les masses .

c - On s'arrête dès que l'on ne peut plus ajouter de masse (dépassement capacité)

```
# ALGORITHME GLOUTON
# Probleme du sac à dos
#
# Algo 1 : classement suivant les masses
# liste_objet est une liste composée de triplets (numero , masse, valeur)
# liste_objet = [ ( 1, masse , valeur ) , ( 2 , masse , valeur ) , .... ]
# par exempe [(1 , 4kg , 8€) , (2 , 5kg , 10€) , (3 , 8kg , 15€)  ...etc ]
#

liste_objet = [(1 , 4 , 8) , (2 , 5 , 10) , (3, 8 , 15) , (4, 3 , 4)]
masse_max = 11

# objet[0] = numero  ,  objet[1] = masse  ,   objet[2] = valeur :

liste_triee = sorted(liste_objet, key=lambda objet: objet[1]) # tri suivant masse

print ("liste triée : ", liste_triee)

masse_totale = 0
objet_pris = []   # liste des numéros des objets pris
valeur_totale = 0

# on parcours la liste triée et on additionne les masses
# si masse totale <= masse_max

for objet in liste_triee:
    if (masse_totale + objet[1]) <= masse_max :
        masse_totale = masse_totale + objet[1]
        objet_pris = objet_pris + [objet[0]]
        valeur_totale = valeur_totale + objet[2]

print ("Masse totale : ",masse_totale , "kg")
print ("Valeur totale : ",valeur_totale , "€")
print ("liste objets pris",objet_pris)
```

Taper et tester le programme.

Vérifier en faisant le calcul sur papier que le programme donne une solution cohérente.

La solution est-elle optimale ? (Rechercher sur papier la solution optimale)

2- Algo2 : Classement suivant les valeurs

Le classement se fait maintenant de la valeur la plus forte à la plus faible :
Modifier la ligne 15 du programme comme suit (remplacer le « ? » par la bonne valeur)

```
liste_triee = sorted(liste_objet, key=lambda objet: objet[ ? ], reverse = True) # tri suivant valeur
```

Notez l'option « reverse » de la fonction de tri python 'sorted' qui permet d'inverser l'ordre de tri pour avoir ici du plus grand au plus petit.
Le critère de tri (key) est une fonction lambda, c'est un peu spécial, nous la verrons plus en détail ultérieurement. Pour le moment, sachez juste que objet ici est juste un paramètre (il pourrait avoir n'importe quel nom)
La suite de l'algorithme ne change pas : Tester et **valider par une vérification sur papier.**

Résultats du programme :

Vos calculs :

Validation prof :

3- Algo 3 : classement en fonction du rapport valeur/masse (du plus fort au plus faible)

```
liste_objet = [(1 , 4 , 8) , (2 , 5 , 10) , (3, 8 , 15) , (4, 3 , 4)]
masse_max = 11

# objet[0] = numero  ,  objet[1] = masse  ,   objet[2] = valeur , objet[3] = valeur/masse:
# on veut créer (en faisant une boucle for) une liste2 qui intègre le rapport valeur/masse:
# de la forme  [ ( 1, masse , valeur , valeur/masse) , ( 2 , masse , valeur , valeur/masse ) , .... ]
# compléter le code ci-dessous et intégrer le dans votre programme au bon endroit :

liste2 = []
for ........ in liste_objet:

    liste2 = liste2 + [  (.......[0],   ......[1],  .......[2],    .......[   ]/......[   ]) ]

print (liste2)

liste_triee = sorted(liste2 , key=lambda objet: objet[....], reverse = True) # tri suivant valeur/masse
```

Vous devriez obtenir après exécution le résultat suivant :

```
[(1, 4, 8, 2.0), (2, 5, 10, 2.0), (3, 8, 15, 1.875), (4, 3, 4, 1.3333333333333333)]
liste triée :  [(1, 4, 8, 2.0), (2, 5, 10, 2.0), (3, 8, 15, 1.875), (4, 3, 4, 1.3333333333333333)]
Masse totale :  9 kg
Valeur totale :  18 €
liste objets pris [1, 2]
```

Vérifier par un calcul la conformité des résultats.
Tester votre programme avec les 19 containers de ti'payet et comparer avec la solution optimale
Masse_max = 31181kg → variable à mettre à jour dans le programme

Valeur masse	1945	4990
	321	1142
2945	4136	10372
7390	1107	3114
1022	1101	3102
2744	962	2624
7280	805	2310
2890	689	2078
1513	1865	4830
3878	13504	32708
9656	667	2034
3926	1833	4766
	16553	40006

Comparez les résultats avec la Solution optimale :
poidsTotal = 30996
valeurTotale = 12248
7390+2744+7280+3926+9656= 30996kg
2945+1022+2890+1513+3878=12248

Conclusion : Ces 3 programmes font partie de la famille des algorithmes **« gloutons »**
Les algos gloutons ne donnent pas (ou rarement) les solutions optimales mais présentent l'intérêt d'être rapide à l'exécution sans utiliser trop de ressources mémoire. Ils sont aussi généralement facile à coder et peuvent convenir comme première approche simple à beaucoup de problèmes.