TD 16 – Algorithmes Gloutons (Greedy)

A / Le problème du sac à dos

Vous avez un **ensemble d'objets** ayant chacun un nom n_i, une valeur v_i et un poids p_i. Vous souhaitez bien entendu maximiser la valeur de votre sac en gardant le poids inférieur à un poids p_{max} maximal, 10 kg par exemple.
Chaque objet peut être représenté par une liste de la forme `objet=(nom,valeur,poids)`. Le nom étant une chaîne de caractère, la valeur un entier de 1 à 10 (10 indiquant un objet très important) et le poids un entier en gramme.

On pourra travailler sur l'ensemble des objets suivants qui sera représenté par une liste de listes :

```
objets=[['book_organic_chemistry',1,1500],['corde',10,4000],['guitare',0,4000],['carte',8,50],['eau',10,1000],['macbook',2,1500],['casque',9,300],['crampons',9,800],['piolet',9,400],['doudou',5,200],['smartphone',3,200],['nourriture',8,2000],['frontale',5,150],['longe',4,400],['trousse_pharmacie',4,500]]
```

Q1 – Ecrire une fonction `tri(objets)` qui, à partir de la liste de listes, renverra la liste de listes `objets_tries` des objets triés par ordre décroissant du rapport valeur/poids. A chaque élément de la liste, on ajoutera l'élément (valeur/poids). Vous devez obtenir la liste de listes suivante :

```
[['carte', 8, 50, 0.16], ['frontale', 5, 150, 0.03333333333333333], ['casque', 9, 300, 0.03], ['doudou', 5, 200, 0.025], ['piolet', 9, 400, 0.0225], ['smartphone', 3, 200, 0.015], ['crampons', 9, 800, 0.01125], ['longe', 4, 400, 0.01], ['eau', 10, 1000, 0.01], ['trousse_pharmacie', 4, 500, 0.008], ['nourriture', 8, 2000, 0.004], ['corde', 10, 4000, 0.0025], ['macbook', 2, 1500, 0.0013333333333333333], ['book_organic_chemistry', 1, 1500, 0.0006666666666666666], ['guitare', 0, 4000, 0.0]]
```

Q2 – Ecrire une fonction `remplir_sac(objets_tries,p_max)` qui, à partir de la liste de listes précédente triée, retournera `(objets_emportes,valeur_sac,poids_sac)` où :
✔ `objets_emportes` est la liste des objets mis dans le sac en optimisant la valeur du sac pour le poids total imposé.
✔ `valeur_sac` est la valeur totale du sac.
✔ `poids_sac` est le poids du sac.
Votre fonction devra utiliser un algorithme glouton évidemment. Vous devez obtenir le résultat suivant pour un poids total de 10 kg maximal :

```
--------------------------------
valeur du sac = 84
poids du sac (en g) = 10000
liste des objets a emporter:
['carte', 'frontale', 'casque', 'doudou', 'piolet', 'smartphone', 'crampons', 'longe', 'eau', 'trousse_pharmacie', 'nourriture', 'corde']
--------------------------------
```


Si vous devez partir faire un sommet en montagne, le résultat obtenu est-il intéressant ? A discuter et méditer !

Q3 – Modifier légèrement le programme pour que le tri s'effectue, de façon décroissante, par la « valeur » de chaque objet et non plus par la « valeur par unité de poids ». Le résultat obtenu est le suivant. Conclusion.

```
----------------------------------
valeur du sac = 84
poids du sac (en g) = 10000
liste des objets a emporter:
['eau', 'corde', 'piolet', 'crampons', 'casque', 'nourriture', 'carte', 'frontale', 'doudou',
'trousse_pharmacie', 'longe', 'smartphone']
----------------------------------
```

B / Végéter

A la télévision, on trouve un ensemble de programmes qui sont tous caractérisés par une liste `(ni,di,fi)` où `ni` est le nom du programme, `di` est l'heure de début du programme et `fi` l'heure de fin. Une personne souhaite regarder le plus grand nombre de programmes possible dans une journée.

Q4 – En utilisant ce qui a été fait dans l'exercice précédent et le paragraphe 3 sur le problème d'organisation, écrire une fonction (qui utilise un algorithme glouton) `vegeter(programmes)` qui renvoie le liste des programmes qu'elle va regarder.