Algorithmes gloutons

1. Rendu de monnaie

On se donne un système de pièces $i \in \{1, 5, 10, 20, 50\}$. L'objectif est de rendre une somme m de monnaie avec le moins de pièces possibles.
On considère bien sûr toutes les valeurs comme étant entières.
Une solution $S = \{i_1, i_2, \ldots, i_p\}$ est un multi-ensemble (il peut contenir des doublons) de pièces.
La solution est optimale si p est minimal.
Une solution gloutonne consiste à choisir la plus grande pièce possible à chaque étape.

Exemple Pour $m = 89$, la solution gloutonne $50, 20, 10, 5, 1, 1, 1, 1$ est optimale.

Remarque La solution gloutonne n'est pas optimale pour tous les systèmes de monnaie.
Prenons par exemple $\{1, 4, 6\}$ et la somme $m = 8$.
La solution gloutonne $\{6, 1, 1\}$ est moins bonne que $\{4, 4\}$.
D'autre part, certains systèmes de pièces empêchent même qu'une solution existe (optimale ou non) quelque soit la somme m à rendre. Si le système contient 1, on est assuré d'avoir une solution (au pire, on ne rend que des pièces 1).

Exercice 1
Écrire une fonction Python pour le rendu de monnaie, prenant en paramètre une liste P pour le système de pièces et une somme m à rendre, et qui utilise la solution gloutonne. Cette fonction retourne une liste solution S, celle des pièces à utiliser pour rendre la monnaie.
Tester alors la fonction pour vérifier son bon fonctionnement sur plusieurs exemples (de systèmes de pièces et de somme à rendre).

2. Un problème de sac à dos

Il existe beaucoup de variantes de problèmes de sac à dos, allant de simple à compliqué dans leur donnée et dans leur résolution, qui n'admettent pas forcément de stratégie gloutonne optimale.
Nous en voyons ici un sous une forme simple, pour lequel la stratégie gloutonne n'est pas optimale.
On dispose de n objets $\{x_1, \ldots, x_n\}$ ayant chacun une valeur v_i et une masse m_i pour $1 \leqslant i \leqslant n$.
On possède un sac à dos ne pouvant supporter qu'une masse maximale M (évidemment, pour que le problème ait un intérêt, la somme totale des n objets, $m_1 + \cdots + m_n$ est supérieure à M).
On cherche le bon choix d'objets à mettre dans le sac à dos de sorte que la valeur totale des objets qu'il contient soit maximale.

Exercice 2
On note m la masse totale des objets choisis, et v la valeur totale de ces objets.
Sachant que chacun des n objets est soit choisi soit non choisi, on peut considérer qu'il existe des nombres c_i, valant respectivement chacun soit 1 soit 0, tels que :

$$m = c_1 m_1 + \cdots + c_n m_n \quad \text{et} \quad v = c_1 v_1 + \cdots + c_n v_n$$

1. Quelles sont les conditions à satisfaire pour m et v ?

2. On souhaite trier les objets par ordre de priorité décroissante.
   Qu'est-ce que cela signifie ?
3. Proposer un algorithme utilisant une stratégie gloutonne, à coder en Python, qui indique les objets à mettre dans le sac.
   La fonction Python prend deux arguments :
   - La liste des objets, sous forme d'une liste de couples (masse, valeur) et déjà triée selon l'ordre établi précédemment.
   - La masse maximale M.

   L'algorithme retourne la liste des c_i, indiquant donc si les objets sont pris ou non, ainsi que le poids mis dans le sac et la valeur totale.
4. Pour tester l'algorithme, on propose la liste suivante d'objets :

   ```
   [(14,126),(2,32),(5,20),(1,5),(6,18),(8,80)]
   ```

   avec un sac à dos de valeur $M = 15$
   On pourra tester différentes manières d'ordonner les objets, et voir les valeurs totales obtenues par la stratégie gloutonne.
   (On rappelle qu'il existe un moyen en Python de trier une liste selon une valeur calculée par une fonction : voir l'activité sur les données en tables)
   La solution optimale, que les diverses manières de trier les objets ne permettent pas d'obtenir par la méthode de résolution gloutonne, a une valeur totale de 132 $(32+20+80)$ pour une masse de $2+5+8=15=M$.