TD Monnayeur glouton MPSI

On veut trouver expérimentalement quelle est la plus grande somme qu'on puisse rembourser de façon optimale dans le système de pièces européen sans utiliser de billet/pièce de 200 euros. On a vu en cours qu'il s'agit de 199 euros mais nous voulons faire une exploration exhaustive d'un arbre de décision pour en avoir le cœur net.

On se donne les variables globales suivantes :

```
#système de pièce européen
coins = [1,2,5,10,20,50,100,200]
"""
nb_max : tableau des nbs max possibles de pièces pour rembourser sans piece 200
         renseigne les contraintes 1 et 2 du cours
exemple : nb_max [1] vaut 2, donc on peut prendre de 0 à 2 pièces de 2 euros
"""
nb_max = [1,2,1,1,2,1,1,0]
```

Question 1.

Proposer une version récursive `monnayeur(v,s)` de la fonction `greedy_change` du cours.

Question 2.

Écrire la fonction

```
"""
respect de la 3eme contrainte
t est un choix de pièces
"""
def constraint3 (t):
```

Le tableau t résume les choix fait : $t[0]$ désigne le nombre de pièces $coins[0]$ choisi, $t[1]$, le nombre de $coins[1]$...

La fonction renvoie renvoie `true` si le tableau t est compatible avec la contrainte 3 du cours.

Question 3.

Écrire la fonction

```
"""
t : un tableau des nb de pièces choisis
coins : un tableau de valeurs de pièces
renvoie : le produit cartésien t . coins
"""
def value(t,coins):
```

On peut voir l'algorithme de recherche de la plus grande somme obtenue sans utiliser de pièce de 200 euros et en respectant les trois contraintes du cours comme l'exploration exhaustive d'un arbre de décision.

À la racine de cet arbre, on choisit le nombre de pièces de valeur 1 utilisées : 0 ou 1, dans les limites imposées par le tableau `nb_max`.

Puis, pour chaque fils de la racine, on choisit le nombre de pièces de valeur 2 : 0, 1 ou 2.

Plus généralement, à chaque nouveau nœud, on choisit le nombre de pièces de la valeur suivante (5, 10, 20, etc.), toujours en respectant les bornes indiquées par le tableau `nb_max`.

On poursuit ainsi l'exploration jusqu'aux feuilles de l'arbre, qui correspondent à des choix complets, c'est-à-dire à un nombre de pièces fixé pour chaque valeur possible. Le respect de la contrainte 3 peut se faire au moement où l'exploration atteint une feuille.

Ainsi, chaque chemin de la racine à une feuille représente une combinaison complète de pièces, autrement dit une solution candidate.

Chaque niveau de l'arbre correspond à une valeur de pièce. Un nœud d'un niveau donné représente un choix partiel ; il engendre l'exploration des choix restant à effectuer, sélectionne la meilleure solution parmi celles-ci et la renvoie à son nœud parent.

Le tableau final obtenu correspond alors à la meilleure combinaison possible, au sens où la somme des valeurs des pièces choisies est maximale.

Question 4.

Écrire la fonction

```
def best(coins,nb_max)
```

Elle prend en paramètre un système de monnaie et un tableau indiquant les contrainte 1 et 2 du cours (`nb_max`). Elle explore l'arbre de décision complet et renvoie la combinaison de pièces respectant les 3 contraintes du cours et permettant d'obtenir une somme maximale sans pièce de 200.

```
best(coins,nb_max) # renvoie   ([0, 2, 1, 0, 2, 1, 1, 0], 199)
```