TD 8 : DES ALGORITHMES GLOUTONS

Exercice 1 (Le rendu de monnaie)

Le problème de rendu de monnaie consiste à rendre la monnaie avec le moins de pièces/-billets possible pour constituer le montant x. On dispose de pièces de monnaie et de billets dont les valeurs en euros sont stockées dans un tableau :

```
valeur = [500,200,100,50,20,10,5,2,1]
```

1. Quelles sont toutes les possibilités pour rendre 7€ ? Existe-t-il une solution optimale (qui permet de rendre le montant avec le moins de pièces/billets possible) ?
2. Écrire une fonction `rendu_monnaie_glouton(valeurs, montant)` qui renvoie un tableau `pieces`, en rendant la monnaie de manière gloutonne : par exemple `[0,0,0,0,0,0,1,0,1]` correspond à 1 pièce de 1€ et 1 billet de 5€.
   Faire un essaie avec 264€. Il doit renvoyer `[0,1,0,1,0,1,0,2,0]`

Exercice 2 (Le problème du sac à dos)

Une question importante à se poser quand on aborde les algorithmes gloutons est de savoir si le meilleur choix fait localement conduit à une solution globale optimale. C'est le but de cet exercice avec le problème du sac à dos.
Votre sac à dos peut porter au maximum 30 kg. Vous avez une liste d'objets que vous pouvez prendre dans votre sac à dos, chaque objet ayant une certaine valeur et un certain poids. Chaque objet ne peut être qu'une seule fois, contrairement aux pièces de monnaie. Comment choisir ces objets pour que le sac à dos ait la plus grande valeur sans dépasser la charge maximale ?

1. Écrire une fonction `sac_a_dos(poids, valeur)` où les tableaus `poids` et `valeur` correspond respectivement au poids/valeur des différents objets. Elle doit renvoyer la valeur du sac à dos et utiliser la stratégie gloutonne suivante consistant à prendre l'objet de plus forte valeur pouvant entrer dans le sac à dos.
2. Faire un essai avec le tableau suivant :

Objets	Poids	Valeur
1	12	7
2	11	4
3	8	3
4	10	3

   La solution trouvée est elle optimale ?
3. Dans le cas suivant, montrer que la solution obtenue n'est pas optimale :

Objets	Poids	Valeur
1	13	7
2	11	4
3	8	3
4	10	3

Exercice 3 (Défi)

On cherche à mettre huits reines sur un échéquier sans que celles-ci ne puissent se prendre en un coup. Écrire une fonction qui affiche les positions des huits dames, de préférence sous la forme d'un échéquier. (on pourra utiliser le module itertools et la fonction permutations : from itertools import permutations ; permutations(range(n)) renvoie toutes les permutations de range(n) (par exemple permutations(range(3)) renvoie [(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)] (par forcément dans cet ordre).