MPSI - ITC - TD 10 : Stratégies gloutonnes

Exercice 1 (Le problème de la grenouille)

On considère n pierres de différentes hauteurs, et un tableau hauteurs qui contient les hauteurs des différentes pierres. Une grenouille démarre sur la pierre d'indice 0 et veut se rendre sur la dernière pierre (d'indice n−1). Si, à un moment donné, la grenouille se trouve sur la pierre d'indice i, elle peut en une étape se rendre sur la pierre d'indice i+1 ou sur la pierre d'indice i+2. Le coût d'un saut pour la grenouille est la valeur absolue de la différence entre la hauteur de départ du saut et la hauteur d'arrivée du saut.

1. Codez une fonction cout_min_grenouille qui prend en entrée un tableau hauteurs décrivant les hauteurs des pierres, et renvoie en sortie un coût possible pour que la grenouille atteigne la dernière pierre depuis la première ; en utilisant une stratégie gloutonne.
2. Si votre fonction est itérative, faites une autre version qui utilise la récursivité. Vous pouvez introduire une fonction auxiliaire qui prend davantage d'arguments.
3. Donnez un tableau hauteurs pour lequel votre fonction ne renvoie pas la solution optimale.

Exercice 2 (Le problème du sac à dos)

On suppose que 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 pris qu'au plus une fois. 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 tableaux `poids` et `valeur` correspond respectivement au poids/valeur des différents objets. Votre fonction proposera un remplissage possible du sac à dos, qui respecte la charge maximale, en suivant la stratégie gloutonne suivante : tant que cela est possible, ajouter au sac à dos l'objet de plus grande valeur ne faisant pas dépasser la charge maximale.
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 (Le rendu de monnaie)

Le problème de rendu de monnaie consiste à rendre la monnaie avec le moins de pièces ou billets possibles, 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 4 (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. Par exemple, permutations(**range**(3)) renvoie une liste de toutes les permutations de (0, 1, 2) (c'est à dire [(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)] , mais pas forcément dans cet ordre).