Algorithmes gloutons

1) Principe

On appelle « algorithme glouton » un algorithme qui suit une méthode de recherche permettant de faire *« le meilleur choix »* à chaque étape de la recherche. Cette méthode ne conduit pas toujours à la solution souhaitée, mais doit donner des résultats acceptables dans un bon nombre de cas…

2) Exemple : le rendu de monnaie

Le problème du rendu de monnaie va nous fournir un exemple simple pour comprendre un algorithme glouton : ***comment décomposer une somme d'argent en utilisant un nombre minimal de pièces à choisir parmi un nombre limité de valeurs possibles ?***
Considérons par exemple les unités monétaires européennes {1, 2, 5, 10, 20, 50, 100, 200, 500} (en €). On suppose disposer d'un nombre illimité de pièces et billets de chaque type, et on souhaite connaître la décomposition minimale en nombre de pièces et billets d'une somme, par exemple $n = 493$ €.

Une **stratégie gloutonne** consiste à rendre la pièce ou le billet v de valeur maximale parmi celles de valeurs inférieures ou égales à n (ici c'est $v = 200$), puis réitérer le procédé avec $n - v$. Sur cet exemple, on obtient : $493 = 200 + 200 + 50 + 20 + 20 + 2 + 1$.

Remarque : Pour certains systèmes de monnaie, l'approche gloutonne peut retourner un résultat optimal, pour d'autres cela ne sera pas le cas. Par exemple, dans le système de pièces (1, 3, 4), l'algorithme glouton n'est pas optimal, comme le montre l'exemple simple suivant. Il donne pour 6 : 4+1+1, alors que 3+3 est optimal.

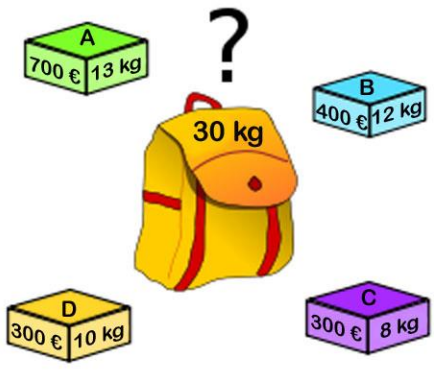
Exercice

Écrire en Python une fonction glouton qui prend en paramètres une somme n et un système de monnaie c supposé trié par ordre croissant et qui retourne la décomposition obtenue suivant la stratégie gloutonne décrite ci-dessus sous la forme d'une liste de valeurs à utiliser.

Par exemple :

```
>>> glouton (493,[1,2,5,10,20,50,100,200,500])
[200, 200, 50, 20, 20, 2, 1]
```

3) Intérêt d'une méthode gloutonne



Le problème du sac à dos : Quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 30 kg autorisés ?

En apparence, la solution la plus simple dans le cas du sac à dos serait d'écrire un algorithme qui teste toutes les combinaisons possibles et qui retient les solutions qui offrent le meilleur gain. Avec seulement 4 objets, cette solution peut être envisagée, mais avec un plus grand nombre d'objets, le temps de calculs, même pour un ordinateur très puissant, deviendrait trop important.

À la place de la méthode « je teste toutes les possibilités », on peut utiliser une méthode gloutonne.

Par exemple, nous ajoutons les objets un par un, chaque ajout d'un objet constitue une étape : à chaque étape on doit choisir un objet à mettre dans le sac.

Il est important de bien comprendre qu'un algorithme glouton ne donne pas forcement une solution optimale.

L'intérêt d'un algorithme glouton est de donner une solution acceptable en un temps acceptable.

Exemple de méthode gloutonne pour la résolution du problème du sac à dos :

Commençons par établir un tableau nous donnant la « valeur massique » de chaque objet :

objet	A	B	C	D
valeur massique	54 €/Kg	33 €/Kg	38 €/Kg	30 €/Kg

On classe ensuite les objets par ordre décroissant de valeur massique : A - C - B -D

Enfin, on remplit le sac en prenant les objets dans l'ordre et en s'arrêtant dès que la masse limite est atteinte.

1re étape : A (13 Kg)

2e étape : C (13+8=21 Kg)

3e étape : B (13+8+12=33 Kg) => impossible, on dépasse les 30 Kg.

Le sac contient 2 objets : A et C pour un montant total de 1000 € et une masse totale de 21 Kg.

La solution trouvée n'est pas optimale. En effet, A et B permet d'atteindre une valeur de 1100 € pour une masse de 25 Kg. Cette méthode gloutonne ne donne pas une solution optimale.