AP09 ALGORITHME GLOUTON

Propriété 1 : Algorithme glouton

Les **algorithmes gloutons** répondent à des problèmes d'optimisation. L'idée est qu'à chaque étape de l'algorithme, celui-ci fasse le choix qui lui semble le plus optimal, sans réfléchir à la suite, en espérant que la solution finale au problème soit elle-même la plus optimale. Attention cela n'est pas forcément toujours le cas et dépend du problème à traiter.

I. Problème de rendu de monnaie

L'exemple classique d'application d'un algorithme glouton est le problème du **rendu de monnaie**. L'objectif est, à partir d'un système de monnaie, de trouver comment rendre une somme de façon optimale, c'est-à-dire avec le moins de pièces et de billets possible.

Exemple

On considère une système monétaire `SYSTEME = [500, 200, 100, 50, 20, 10, 5, 2, 1]`. La question est de savoir comment rendre une somme particulière. Dans un premier temps, on renverra une liste des billets ou pièces à rendre. On peut de façon naïve déployer cet algorithme :

1. Tant que la somme à rendre est supérieure à 0
2. Si la somme à rendre est plus grande que la pièce observée alors :
   - on soustrait cette pièce à la somme à rendre
   - insère la valeur de la somme à rendre dans la liste de choix
3. Sinon on prend la pièce de valeur immédiatement inférieure et on reprend à la ligne 1

Exercice 1

1. Implémenter cette fonction en python

Code Python

```
1  def rendre_monnaie1(systeme:list, somme:int)->list:
2      ''' Renvoie la liste des pièces à rendre '''
3      i = 0                       # Indice de la pièce actuelle
4      choix = []                   # Liste de choix
5      while somme > 0:
6          if somme >= SYSTEME[i]:
7              somme -= SYSTEME[i]
8              ...
9          else:
10             ...
11     return choix
12
13 SYSTEME = [500, 200, 100, 50, 20, 10, 5, 2, 1]
14 assert rendre_monnaie1(SYSTEME, 674) == [500, 100, 50, 20, 2, 2]
```

2. Compléter la trace de l'algorithme pour `somme = 674`

somme	i	choix	while	SYSTEME[i]	IF	somme	choix	i
674	0	[]	True	500	True	174	[500]	0
174	0	[500]						

3. Modifier cet algorithme en prenant un système sous forme de listes de listes. Chaque sous liste contient la valeur du billet ou de la pièce et le nombre disponible.

   Par exemple : `SYSTEME2 = {500: 1, 200: 1, 100: 1, 50: 2, 20: 2, 10: 2, 5: 3, 2: 6, 1: 4}`

   Attention, il faudra vérifier que le système permet de rendre la somme à rendre

Remarque

Le système de monnaie a été choisi pour qu'un algorithme glouton soit optimal. Comment payer 9 € avec un système de pièces ou de billets de 1 €, 2 €, 5 € et 10 €. Le choix optimal (moins de pièces et de billet) s'obtient en appliquant l'algorithme glouton (choisir toujours la plus grande des valeurs pour compléter) soit 5 € + 2 € + 2 € (3 pièces). Imaginons qu'il existe un billet de 7 €. L'algorithme glouton n'est plus optimal. En effet si l'on souhaite payer 14 €, il donnerait : 10 € + 2 € + 2 € alors que le meilleur choix serait 7 € + 7 €.

II. Problème du sac à dos

Le problème du sac à dos (aussi noté KP pour *knapsack problem*) est lui aussi un problème d'optimisation.

On dispose de plusieurs objets (chaque objet possède une valeur et un poids). Seulement ce sac ne peut supporter plus d'un certain poids. On va donc devoir choisir des objets et maximiser la valeur totale dans le sac sans toutefois dépasser le poids maximum.

On suppose pour la suite de l'exercice que le sac ne peut supporter que 15 kg maximum

1. Méthode naïve ou exhaustive

Une première méthode consiste à lister toutes les combinaisons possibles d'objets puis de regarder lesquelles ne dépassent pas 15 kg et enfin de regarder laquelle permet d'avoir une somme maximale.

S'il y a n objets, alors il y a 2^n combinaisons possibles. La complexité est donc en $O(2^n)$, c'est un coût exponentiel.

2. Méthode glouton

Un algorithme glouton fait le meilleur choix possible à chaque étape en espérant que la solution finale soit elle-même la meilleure.

Voici cinq objets, leur poids et leur valeur. Vous disposez d'un sac de capacité maximale de 32 kg.

Exercice 2

1. Choisissons un critère de choix d'un objet à mettre dans le sac. Quelle est la valeur du sac si on applique ce critère à chaque choix d'un objet ?
2. Définissez un autre critère de choix. Appliquez-le à chaque choix d'objet et donnez la valeur obtenue
3. Même question avec un troisième critère de choix.
4. Y a-t-il une meilleure valeur totale ?

Objet	Poids (kg)	Valeur (euros)
A	16	512
B	15	495
C	19	950
D	14	280
E	14	420

Définition 1 : Algorithme glouton

Type d'algorithme dans lequel, à chaque étape, un **choix optimal local** est fait, sans tenir compte de la suite et sans possibilité de retour en arrière (pensez à un glouton qui avale un objet, impossible de revenir en arrière...). On peut donc comparer :

- la méthode **exhaustive** (étude de tous les cas ou force brute) : permet de trouver la meilleure solution à un problème mais en un temps qui peut être considérable ;
- une méthode **gloutonne** : ne donne pas forcément la solution optimale à un problème mais donne souvent une solution acceptable en un temps limité. En ce sens, une méthode gloutonne est heuristique, c'est-à-dire une méthode qui permet de trouver une solution convenable mais non forcément optimale à un problème mais en un temps raisonnable en évitant une analyse complète de ce problème.