Algorithmes gloutons

Lorsque l'on conçoit un algorithme pour résoudre un problème, il faut choisir une stratégie, ce que l'on appelle un paradigme.

Parmi les différents paradigmes, il existe la stratégie gloutonne que l'on va essayer de présenter ici.

Une stratégie gloutonne consiste à opérer le meilleur choix à chaque étape de la résolution du problème, en espérant obtenir une solution globale optimisée.

Pour prouver la validité d'un algorithme glouton, on peut chercher un contre-exemple qui met en échec la stratégie adoptée. Si l'on n'en trouve pas, on peut utiliser une preuve de corrections, ce qui n'est pas demandé ici.

Exemples :

Le problème du sac à dos

Voici le contexte : On possède un sac à dos dont on connaît la capacité maximale en kilo. Il s'agît de le remplir avec des objets ayant des poids différents et des valeurs différentes. Le but est d'optimiser la valeur du sac à dos.

1. Votre sac à dos supporte 7 kilos. Vous avez 10 livres, chacun ayant une valeur de 8 € et pesant 1 kilo, deux bouteilles de 2 kilos à 10 € l'unité .
   a) Optimisez le contenu du sac à dos pour qu'il ait une valeur maximale.
   b) On ajoute une pierre de 4 kilos d'une valeur de 12 € et une autre d'un poids de 2.5 kilos et d'une valeur de 15 €. Optimisez le contenu du sac
2. On imagine bien que le problème peut devenir complexe.

On va considérer des couples d'objets, le premier paramètre étant la valeur totale de l'objet et la seconde la quantité présente. On va considérer que l'on peut remplir le sac à dos avec une partie de l'objet (5 kilos de riz par exemple sur les 15 à notre disposition).

On a à notre disposition :

Poudre d'or(10000,10), cuivre(5000,100), émeraude (2000,10), fer(3000,150).

Notre sac supporte 35 kilos.

Détermination de la stratégie.

Ici, on va calculer le prix au kilo, classer les ingrédients du plus cher au moins cher, et remplir le sac en commençant par l'ingrédient le plus cher jusqu'à atteindre la contenance du sac ou l'épuisement des denrées.

On obtiendrait : poudre d'or(1000),émeraude (200),cuivre(50), fer(20).

A partir de là, on remplit le plus possible avec de la poudre d'or, puis l'émeraude…

On aurait alors 10 kilos de poudre d'or,10 kilos d'émeraude et 10 kilos de cuivre .

Implémentation de cet algorithme

Implémentation d'un algorithme glouton

Le problème du sac à dos.

On rappelle le contexte : Il s'agît de remplir un sac à dos dont le poids maximum est connu avec des objets dont on connaît le couple (valeur, poids). On a vu que le critère de choix était le prix au kilo.

A partir d'un dictionnaire donné, écrire une fonction qui classe les objets par prix décroissants au kilo.

On utilisera, une fois le tri fait, l'instruction *reserve* = *True*

Entrée [20]:
```
def sac_a_dos(dico):
    tri = sorted(dico.items(), key= "todo"

    return tri
sac_a_dos({"a": (120, 10), "b": (300, 20), "c": (120, 5)})
assert (sac_a_dos({"a": (120, 10), "b": (300, 20), "c": (120, 5)}))\
        == [('c', (120, 5)),('b', (300, 20)), ('a', (120, 10))]
```

Créer une fonction qui à une liste triée suivant les conditions du problème renvoie le sac à dos optimal. Dans cette fonction on associera :

- à la variable nom, le nom de l'objet
- à la variable poids, le poids de l'objet

On rappelle qu'un élément de la liste a cette structure ('a',(120,10))

Entrée [ ]:
```
def sac_a_dos_optimal(liste,poids_max):

    solution = []
    for o in liste:
        nom = #todo
        poids = #todo
        solution.append(#todo)
        poids_max = poids_max- #todo
        if poids_max #todo :
            break
    return solution
assert (sac_a_dos({"a": (120, 10), "b": (300, 20), "c": (120, 5)}, 30))\
        == [("c", 5), ("b", 20), ("a", 5)]
```

Il ne reste plus qu'à finaliser en collant les morceaux.

Entrée [ ]:
```
# Créer un dictionnaire de votre choix contenant 5 objets, chacun associé à un couple (valeur, poids)
dico ={ "...":(valeur, poids), "..." :.....}
liste_triee = #todo
sac_a_dos_optimal(#todo)
```

Le rendu de monnaie

Ici, un commerçant veut optimiser son rendu de monnaie. Il veut rendre le moins de pièces et de billets possibles.

Il possède en quantité suffisante dans le cadre de l'exercice des billets de 20 et 10 € ainsi que toutes les pièces (2,1,0.5,0.2,0.1,0.05,0.02,0.01).

- Proposer une stratégie gloutonne.
- Essayez sur quelques exemples de trouver un contre-exemple à votre stratégie :
  Rendre 2.35 €, 53 €. Rendre la monnaie avec un prix de 0.88 cents payé avec un billet de 20 €.
- Ecrire un algorithme, en précisant les données en entrées, et l'attendu en sortie.
- Implémentez cet algorithme en langage python, tout en proposant dans votre programme une assertion.