

http://www.info-mounier.fr/premiere_nsi/algorithmique/index.php

Algorithmes gloutons

1. Présentation de la notion d'algorithme glouton

Les problèmes d'optimisation

L'**optimisation** est une branche des mathématiques cherchant à **modéliser**, à **analyser** et à **résoudre** les problèmes qui consistent à **minimiser** ou **maximiser** une fonction sur un ensemble.

Les **problèmes d'optimisation** classiques sont par exemple :

- la **répartition optimale de tâches** suivant des critères précis (emploi du temps avec plusieurs contraintes) ;
- le problème du **rendu de monnaie** ;
- le problème du **sac à dos** ;
- la recherche d'un **plus court chemin dans un graphe** ;
- le problème du **voyageur de commerce**.

Résoudre un problème d'optimisation : les algorithmes gloutons

De nombreuses techniques informatiques sont susceptibles d'apporter une solution exacte ou approchée à ces problèmes.

- **Recherche de toutes les solutions**
La technique la plus basique pour résoudre ce type de problème d'optimisation consiste à **énumérer de façon exhaustive toutes les solutions possibles**, puis à choisir la meilleure. Cette approche par **force brute**, impose souvent un coût en temps trop important pour

être utilisée.

- **Les algorithmes gloutons**

Un **algorithme glouton** (*greedy algorithm*) est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local.

Au cours de la construction de la solution, l'algorithme résout une partie du problème puis se focalise ensuite sur le sous-problème restant à résoudre.

La **méthode gloutonne** consiste à choisir des solutions locales optimales d'un problème dans le but d'obtenir une solution optimale globale au problème.

- Le principal avantage des algorithmes gloutons est leur **facilité de mise en œuvre**.
- Le principal défaut est qu'ils ne renvoient **pas toujours la solution optimale** nous le verrons.
- Dans certaines situations dites **canoniques**, il arrive qu'ils renvoient non pas un optimum mais l'optimum d'un problème

2. Le problème du rendu de monnaie

Un achat dit en espèces se traduit par un échange de pièces et de billets. Dans la suite, les pièces désignent indifféremment les véritables pièces que les billets.

Supposons qu'un achat induise un rendu de 49€ en considérant pour simplifier que les 'pièces' prennent les valeurs 1, 2, 5, 10, 20, 50, 100 euros. Quelles pièces peuvent être rendues ?

- $49=4\times 10+1\times 5+2\times 2=4\times 10+1\times 5+2\times 2$ soit 7 pièces au total (Quatre pièces de 10 euros, 1 pièce de 5 euros et deux pièces de 2 euros.)
- ou $49=9\times 5+2\times 2=9\times 5+2\times 2$ soit 11 pièces au total
- ou $49=9\times 5+4\times 1=9\times 5+4\times 1$ soit 13 pièces au total
- ou $49=49\times 1=49\times 1$ soit 49 pièces au total

Le **problème du rendu de monnaie** consiste à déterminer la solution avec le **nombre minimal de pièces**.

Rendre 49 euros avec un minimum de pièces est un problème d'optimisation. En pratique, tout individu met en œuvre un algorithme glouton.

1. Il choisit d'abord la plus grande valeur de monnaie, parmi 1, 2, 5, 10, contenue dans 49 euros.
En l'occurrence, quatre fois une pièce de 10 euros.
2. La somme de 9 euros restant à rendre, il choisit une pièce de 5 euros,
3. Puis deux pièces de 2 euros.

Algorithme Glouton : le rendu de monnaie

Système = { 1 ; 2 ; 5 ; 10 ; 20 ; 50 ; 100 }

Valeur = 49 euros

	Choix de la 'pièce' du système monétaire	
Etape 1	20	reste à rendre : $49 - 20 = 29$
Etape 2	20	reste à rendre : $29 - 20 = 9$
Etape 3	5	reste à rendre : $9 - 5 = 4$
Etape 4	2	reste à rendre : $4 - 2 = 2$
Etape 5	2	reste à rendre : $2 - 2 = 0$

$$49 = 20 + 20 + 5 + 2 + 2$$

Solution optimale et système canonique du rendu de monnaie

Cette stratégie gagnante pour la somme de 49 euros l'est-elle pour n'importe quelle somme à rendre ?

- **Un système canonique**
 - On peut montrer que l'**algorithme glouton du rendu de monnaie** renvoie une solution optimale pour le système monétaire français {1,2,5,10,20,50,100}

- Pour cette raison, un tel système de monnaie est qualifié de **canonique**.

- **Des systèmes non canoniques**

- D'autres systèmes ne sont pas canoniques. L'algorithme glouton ne répond alors pas de manière optimale.
- Par exemple, avec le système {1,3,6,12,24,30}, l'algorithme glouton répond en proposant le rendu : $49 = 30 + 12 + 6 + 1 = 49$ soit 4 pièces alors que la solution optimale est $49 = 2 \times 24 + 1 = 49$, soit 3 pièces.

3. Le TD sur les algorithmes gloutons et le rendu de monnaie

On cherche à implémenter un algorithme glouton de **rendu de monnaie** en ne considérant que des valeurs entières des pièces du système. Par ailleurs, la plus petite pièce du système sera toujours 1, on est ainsi certain de pouvoir rendre la monnaie quelque soit la somme choisie.

Fonctionnement de l'algorithme glouton du rendu de monnaie

1. On choisit un **système** de monnaies, par exemple : $\text{système} = \{1, 2, 5, 10, 20, 50, 100\}$
2. On choisit une **valeur**, par exemple $\text{valeur} = 49$ euros .
3. On choisit la plus grande 'pièce' du système inférieure à la valeur et on l'ajoute à la liste des pièces à rendre.
4. On calcule le reste à rendre et on recommence jusqu'à obtenir un reste à rendre nul.

Algorithme Glouton : le rendu de monnaie

Système = { 1 ; 2 ; 5 ; 10 ; 20 ; 50 ; 100 }

Valeur = 49 euros

	Choix de la 'pièce' du système monétaire	
Etape 1	20	reste à rendre : $49 - 20 = 29$
Etape 2	20	reste à rendre : $29 - 20 = 9$
Etape 3	5	reste à rendre : $9 - 5 = 4$
Etape 4	2	reste à rendre : $4 - 2 = 2$
Etape 5	2	reste à rendre : $2 - 2 = 0$

$$49 = 20 + 20 + 5 + 2 + 2$$

Exercice 1

1. Ecrire une fonction python qui reçoit deux arguments – la somme à rendre et le système de monnaie – et qui renvoie la liste des pièces choisies par l'algorithme glouton.
2. Tester votre fonction avec les valeurs et les systèmes proposés dans les exemples du cours ci-dessus.
3. Inventez un **système non canonique** différent de celui de l'exemple (mais toujours de valeur minimale 1) et trouver un exemple qui le prouve.
Votre fonction devra donc donner une solution mais qui n'est pas optimale.
4. Complément : créer un programme (ou une autre fonction) qui va afficher toutes les informations suivantes :
 $49 = 4 \times 10 + 1 \times 5 + 2 \times 2$
Soit 7 pièces au total
C'est à dire : Quatre pièces de 10 euros, 1 pièce de 5 euros et deux pièces de 2 euros.

Exercice 2

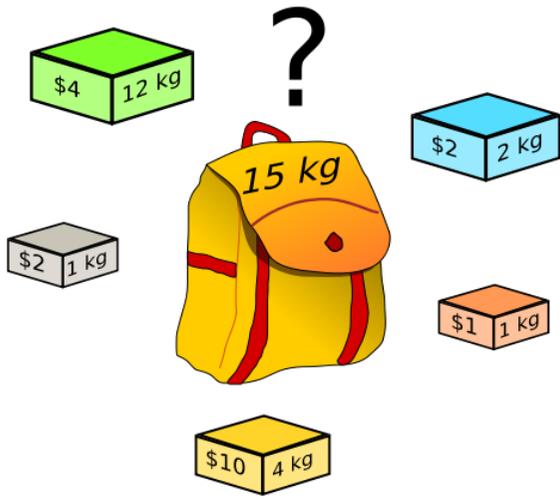
- On cherche maintenant à généraliser notre algorithme avec le système de pièces et de billets utilisées en Europe et des valeurs décimales.
 - On va donc considérer un système composé de pièces et de billets :
 - Les pièces (en euros) : 0,01 - 0,02 - 0,05 - 0,10 - 0,20 - 0,50 - 1 - 2 ;
 - Les billets (en euros) : 5 - 10 - 20 - 50 - 100 - 200 - 500.
1. Modifier donc votre programme afin qu'il affiche le nombre des pièces à rendre et les billets.
 2. Tester les avec plusieurs exemples.
 3. Et si il n'y avait ni billet de 5, ni billet de 10 euros.
Montrer avec un exemple bien choisi que la solution donnée par l'algorithme n'est pas optimale

Exercice 3 - Une pièce de 7 euros

- On peut se demander pourquoi il n'y a pas de pièce de 7 euros par exemple.
 - En fait c'est parce que si tel était le cas, l'algorithme glouton de rendu de monnaie ne serait plus optimal.
1. Tester votre algorithme en ajoutant une pièce de 7 euros dans le système.
 2. Trouver un exemple qui renvoie une solution qui n'est pas optimale.

4. Le problème du sac à dos

Le **problème du sac à dos**, noté également **KP** (en anglais, **Knapsack Problem**) est un problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.



Remarque historique

Le problème du sac à dos est l'un des **21 problèmes NP-complets** de Richard Karp, exposés dans son article de 1972. Il est intensivement étudié depuis le milieu du XXe siècle et on trouve des références dès 1897, dans un article de George Ballard Mathews. La formulation du problème est fort simple, mais sa résolution est plus complexe.

Description du problème du sac à dos et stratégies

On dispose d'un sac pouvant supporter un poids maximal donné et de divers objets ayant chacun une valeur et un poids. Il s'agit de choisir les objets à emporter dans le sac afin d'obtenir la valeur totale la plus grande tout en respectant la contrainte du poids maximal.

Ce problème peut se résoudre par force brute, c'est-à-dire en testant tous les cas possibles. Mais ce type de résolution présente un problème d'efficacité. Son coût en fonction du nombre d'objets disponibles croît de manière exponentielle.

Nous pouvons envisager une **stratégie gloutonne**. Le principe d'un algorithme glouton est de faire le meilleur choix pour prendre le premier objet, puis le meilleur choix pour prendre le deuxième, et ainsi de suite. Plusieurs stratégies sont possibles :

- Prendre l'objet qui a la plus grande valeur ;
- Prendre l'objet qui a le plus petit poids ;
- Prendre l'objet qui a le rapport valeur/poids le plus grand.

1 Énoncé général

On dispose de plusieurs objets possédants chacun un poids et une valeur, et d'un sac à dos acceptant un poids maximum. Quels objets faut-il mettre dans le sac à dos de manière à maximiser la valeur totale sans dépasser le poids maximum autorisé ? On souhaite répondre au problème en utilisant un algorithme glouton.

1. On va donc considérer un critère global : le poids du sac à dos qui ne doit pas dépasser une certaine valeur.

2. On doit choisir aussi un des critères locaux que l'on choisit au départ pour résoudre le problème :

- Prendre l'objet qui a la plus grande valeur;
- OU Prendre l'objet qui a le plus petit poids;
- OU Prendre l'objet qui a le rapport valeur/poids le plus grand.

2 Un cas particulier

On possède 4 objets dont les valeurs et masses respectives sont données ci-contre :

objet n°	1	2	3	4
valeur (en €)	7	4	3	3
masse (en kg)	13	12	10	8

Si l'on dispose d'un sac à dos pouvant accepter une masse maximale de 30 kg, quels objets doit-on choisir ? Quelle est la valeur totale ? Quelle est la masse totale contenue dans le sac ?

1. Écrire une fonction `sac_a_dos` prenant en paramètre la masse maximale du sac à dos et une liste de couple (valeur, poids). Cette fonction renvoie la liste des objets, la valeur totale des objets et la masse du sac à dos. On prendra comme critère local, de prendre l'objet qui a la plus grande valeur ;

La liste des objets

```
liste_objets=[[7,13],[4,12],[3,10],[3,8]]
```

Exemple : `>>> sac_a_dos(15,liste_objets) ([[7, 13]], 7, 13)`

```
>>> sac_a_dos(22,liste_objets) ([[7, 13], [3, 8]], 10, 21)
```

2. Écrire une deuxième fonction `sac_a_dos2` qui met en oeuvre le deuxième critère : Prendre l'objet qui a le plus petit poids.

```
>>> sac_a_dos2(15,liste_objets) ([[3, 8]], 3, 8)
```

```
>>> sac_a_dos2(22,liste_objets) ([[3, 8], [3, 10]], 6, 18)
```

3. Quelles sont les réponses obtenues à l'aide des deux fonctions pour une masse maximale de 30 kg ? L'une d'elles est-elle optimale ? On pourra utiliser un arbre pour déterminer toutes les solutions possibles.

4. Essayer avec les listes

(a) `liste_objet=[[4, 11], [4, 1], [5, 5], [7, 14], [13, 6], [14, 13], [14, 4], [10, 9], [6, 13], [9, 4], [4, 9], [7, 8], [6, 6], [9, 6], [8, 5], [11, 12], [6, 13], [8, 11], [6, 13]]`

La meilleure solution est :

```
([[4, 1], [13, 6], [14, 4], [10, 9], [9, 4], [9, 6]], 59, 30)
```

(b) `liste_objet=[[11, 9], [1, 1], [14, 1], [9, 1], [13, 13], [1, 12], [9, 14],[14, 10], [13, 6], [4, 5], [12, 1], [12, 2], [13, 12], [14, 15],[9, 3]]`

La meilleure solution est :

```
([[1, 1], [14, 1], [9, 1], [14, 10], [13, 6], [4, 5], [12, 1], [12, 2], [9, 3]], 88, 30)
```

Algorithmes gloutons - EXERCICES

Exercice 1

On cherche à sélectionner cinq nombres de la liste suivante en cherchant à avoir leur somme la plus grande possible (maximiser une grandeur) et en s'interdisant de choisir deux nombres voisins (contrainte).

15-4-20-17-11-8-11-16-7-14-2-7-5-17-19-18-4-5-13-8

Comme on souhaite avoir le plus grand résultat final, la stratégie gloutonne consiste à choisir à chaque étape le plus grand nombre possible dans les choix restants.

1. Appliquez cet algorithme glouton sur le tableau.
2. Vérifiez que $\{20,18,17,16,15\}$ est une autre solution possible.
3. Que dire de la solution gloutonne ?

Exercice 2 : le problème du voyageur

	Nancy	Metz	Paris	Reims	Troyes
Nancy		55	303	188	183
Metz	55		306	176	203
Paris	303	306		142	153
Reims	188	176	142		123
Troyes	183	203	153	123	

Un voyageur a ciblé plusieurs villes qu'il souhaite visiter. Il cherche un itinéraire passant par toutes ces villes et qui minimise la distance totale parcourue. Les villes peuvent être visitées dans n'importe quel ordre mais aucune ne doit être négligée, et le visiteur doit revenir à la fin à sa ville de départ.

Le voyageur part de Nancy et souhaite visiter Metz, Paris, Reims et Troyes, avant de retourner à Nancy.

Voici un tableau donnant les distances kilométriques entre chacune de ces villes.

1. Quelle est la stratégie gloutonne à mettre en œuvre ?
2. Mettez en œuvre cette stratégie et donnez la solution.
3. Calculez la distance totale pour le parcours Metz - Reims - Paris - Troyes (départ et arrivée à Nancy sous-entendus)
4. Que dire de la solution gloutonne ?

Exercice 3 : le problème du sac à dos

On rappelle qu'il y a plusieurs stratégies gloutonnes pour donner une solution à ce problème

- **Stratégie 1** : prendre toujours l'objet de plus grande valeur n'excédant pas la capacité restante (il faut trier préalablement par valeur décroissante)
- **Stratégie 2** : prendre toujours l'objet de plus faible masse (il faut trier préalablement par masse croissante)
- **Stratégie 3** : prendre toujours l'objet de plus grand rapport valeur/masse n'excédant pas la capacité restante (il faut trier préalablement par rapport valeur/masse décroissant)

Exemple 1

Considérons les objets suivants et un sac de capacité maximale 2 kg.

objet	A	B	C
masse (kg)	1,5	2	0,3
valeur (€)	200	500	400
Valeur/masse	133,33...	250	1333,33...

1. Appliquez chacune des stratégies à ce problème.
2. Quelle est la meilleure stratégie dans ce cas ?
3. Listez toutes les combinaisons possibles, puis celles respectant la contrainte et déduisez-en la solution optimale au problème. Comparez-la aux stratégies gloutonnes.

Exemple 2 Même questions mais avec un sac de capacité maximale 3,5 kg.

Exemple 3 On considère désormais les objets suivants et un sac de capacité maximale 5 kg.

Objet	Valeur (en milliers d'€)	Masse (en kg)
A	114	4.57
B	32	0.63
C	20	1.65
D	4	0.085
E	18	2.15
F	80	2.71
G	5	0.32

1. Appliquez chacune des stratégies à ce problème.
2. Quelle est la meilleure stratégie dans ce cas ?
3. Le sac $\{B,C,F\}$ est-il une solution au problème ? Quelle est sa valeur ? Que dire des solutions gloutonnes ?

Exercice 4

Vous visitez un parc d'attractions proposant des spectacles à différents horaires. Voici les horaires des différents spectacles :

spectacle	A	B	C	D	E	F	G	H	I	J
horaire	10h-11h	10h30-11h30	11h-12h30	11h30-12h	12h-13h	13h-15h	13h30-14h	14h-15h30	15h-16h	16h-17h30

Vous avez remarqué qu'il n'est pas possible d'assister à tous les spectacles puisque certains ont lieu à des moments communs. Vous souhaitez assister à un maximum de spectacles sur la journée. Quels spectacles devez-vous choisir ?

Voici deux stratégies gloutonnes possibles :

- Stratégie n°1 : choisir le spectacle dont l'heure de début arrive le plus tôt (parmi les spectacles dont l'heure de début est postérieure aux créneaux des spectacles déjà choisis). Cette stratégie est basée sur l'idée que moins on attend entre deux spectacles, plus on en verra.
- Stratégie n°2 : choisir le spectacle dont l'heure de fin arrive le plus tôt (parmi les spectacles dont l'heure de début est postérieure aux créneaux des spectacles déjà choisis). Cette stratégie est basée sur l'idée que plus un spectacle finit tôt, plus il reste de temps pour en voir d'autres.

1. Appliquez ces deux stratégies au problème.
2. Laquelle donne la meilleure solution ?

Remarque : la deuxième stratégie gloutonne donne toujours la solution optimale à ce type de problème.