

# Chapitre 11

## Les algorithmes gloutons

En informatique, on rencontre souvent des problèmes d'optimisation comme celui mis en œuvre dans le rendu de monnaie. Les algorithmes gloutons sont couramment utilisés pour les résoudre.

### 11.1

#### Le rendu de monnaie

NSI 1ÈRE - JB DUTHOIT

##### 11.1.1 Le problème

On veut programmer une caisse automatique pour qu'elle rende la monnaie de façon optimale, c'est-à-dire avec le nombre minimal de pièces et billets.

La valeur des pièces et billets à disposition sont : 1, 2, 20, 5, 10, 50, 100 et 200 euros.

On dispose d'autant de pièces et de billet que l'on le souhaite.

Exemple : Louis veut acheter un objet qui coûte 53 euros. Elle paye avec un billet de 100 euros.

La caisse automatique doit lui rendre 47 euros.

La meilleure façon de rendre la monnaie est de le faire avec deux billets de 20, un billet de 5 et une pièce de 2 euros.

##### 11.1.2 Résolution du problème par la force brute

La solution naïve consiste à énumérer toutes les solutions possibles puis choisir la solution optimale, celle qui minimise le nombre de pièces et de billets. On peut totaliser 47 euros de différentes façons, par exemple :

Rendus de monnaie	Nombre de pièces et billets
$47 \times 1 \text{ €}$	47
$45 \times 1 + 1 \times 2 \text{ €}$	46
$6 \times 1 + 3 \times 2 + 3 \times 5 + 2 \times 10$	14
$7 \times 1 + 4 \times 10$	11

Cette méthode permet de trouver la solution optimale globale au problème, mais le nombre de possibilités est très important. L'utilisation d'un tel algorithme ici est très coûteux en temps de calcul.

### 11.1.3 L'algorithme glouton

Le principe de l'algorithme glouton :

Utiliser un algorithme glouton consiste à optimiser la résolution d'un problème en utilisant l'approche suivante : **on procède étape par étape, en faisant, à chaque étape, le choix qui semble le meilleur, sans jamais remettre en question les choix passés.**

Application au rendu de monnaie :

Dans l'exemple du problème de rendu de monnaie, on commence par rendre la pièce ou le billet de la plus grande valeur possible, c'est-à-dire dont la valeur est inférieure à la somme à rendre. On déduit alors cette valeur de la somme à rendre, ce qui conduit à un problème de plus petite taille. On recommence ainsi jusqu'à obtenir une somme nulle.

**Données : la somme à rendre et la liste des pièces et billets à disposition.**

**Résultat : une liste des pièces et billets à rendre.**

Algorithme :

```
initialiser monnaie à la liste vide ;
initialiser somme_restante à somme ;
tant que la somme_restante est strictement positive :
on choisit dans la liste la plus grande valeur qui ne dépasse pas la somme restante
on ajoute cette valeur à monnaie,
somme_restante = somme_restante - valeur_choisie ;
renvoyer monnaie
```

### 11.1.4 Solution du problème de rendu de monnaie

Dans notre exemple on a :

```
somme = 47
liste = [1, 2, 5, 10, 20, 50, 100, 200]
```

L'algorithme permet de résoudre le problème étape par étape :

```
Étape monnaie montant restant
Initialisation monnaie = [] somme_restante = 47
Étape 1 monnaie = [20] somme_restante = 27
Étape 2 monnaie = [20, 20] somme_restante = 7
Étape 3 monnaie = [20, 20, 5] somme_restante = 2
Étape 4 monnaie = [20, 20, 5, 2] somme_restante = 0
Résultat : [20, 20, 5, 2]
```

⚠️ Un algorithme glouton permet de trouver une solution optimale locale au problème, mais pas toujours une solution optimale globale.

Par exemple, si notre caisse automatique doit rendre une somme de 63 € mais qu'elle ne dispose plus de billets de 5 euros ni de 10 euros.

```
somme = 63
liste = [1, 2, 20, 50, 100, 200]
```

L'algorithme glouton donne comme résultat `resultat = [50, 2, 2, 2, 2, 2, 2, 1]` alors que la solution optimale globale est `solution_optimale_globale = [20, 20, 20, 2, 1]`

**Exercice 11.184**

Implémenter cet algorithme en Python : créer une fonction `rendu_monnaie(somme,pièce)` avec comme paramètres `somme` (la somme à rendre) et `pièce` la liste des pièces disponibles rangées dans l'ordre décroissant.