version pdf

Synthèse du cours Algorithmes gloutons

Solution approchée d'un problème d'optimisation et heuristique gloutonne

> **"Heuristique gloutonne et algorithme glouton"**
>
> Une *heuristique* est une méthode de résolution approchée d'un problème.
>
> Face à un problème d'optimisation, une heuristique qui procède par une succession de choix localement optimaux pour construire une solution globalement optimale, en ne remettant jamais en cause les choix précédents, s'appelle une **heuristique gloutonne**.
>
> Un algorithme qui construit une solution avec une heuristique gloutonne est un **algorithme glouton**.

Caractéristiques d'un algorithme glouton

> **"Caractéristiques d'un algorithme glouton"**
>
> - Un *algorithme glouton construit une solution par une succession de choix localement optimaux*, en espérant obtenir une solution globalement optimale
> - Un *algorithme glouton est efficace*, car il progresse directement vers une solution sans jamais remettre en question les choix précédents. Sa complexité est souvent facile à établir. Le travail pour effectuer chaque *choix glouton* est souvent effectué lors d'un prétraitement, par exemple avec un tri de l'entrée.
> - Un *algorithme glouton n'est pas toujours correct*, et s'il l'est, la preuve peut être difficile.

Un algorithme glouton optimal dans certains cas

"Problème du rendu de monnaie"

On se place dans la position du caissier qui doit rendre en monnaie un certain montant avec un nombre minimal de pièces. On suppose que le caissier dispose en nombre illimité de toutes les valeurs de pièces disponibles. L'ensemble des valeurs de pièces disponibles constitue le *système monétaire*.

Il s'agit d'un **problème d'optimisation** dont la spécification est la suivante :

- **Entrée du problème :** un montant à rendre et une liste de valeurs de pièces d'un système monétaire ; on suppose qu'on dispose d'un nombre illimité de pièces de chaque valeur
- **Sortie du problème :** une liste de pièces dont la somme est égale au montant à rendre et dont le nombre de pièces est minimal ; ou une liste vide si le montant ne peut être atteint avec les pièces du système

"Algorithme glouton de rendu de monnaie"

La recherche exhaustive d'une solution n'est pas raisonnable : il faudrait déterminer toutes les décompositions en somme de pièces de la monnaie à rendre. Une heuristique gloutonne de construction d'une solution est assez naturelle et peut se résumer en une phrase :

> - tant qu'il reste un montant à rendre on choisit la plus grande valeur de pièce disponible inférieure ou égale à la somme et on la retranche du montant à rendre

Cet algorithme glouton se termine dès que le système monétaire contient une pièce de valeur 1.

L'optimalité de l'algorithme glouton dépend du système monétaire. Par exemple avec le système monétaire `[1, 4, 6]` l'algorithme glouton rend le montant 8 avec les pièces `[6, 1, 1]` ce qui n'est pas optimal car on peut rendre en deux pièces avec `[4, 4]`.

Cependant il existe des systèmes monétaires, dits *canoniques*, où l'algorithme glouton rend toujours la monnaie en un nombre minimum de pièces. On ne connaît pas de critère simple pour déterminer si un système est canonique mais on peut démontrer que le système de l'euro est canonique. Par ailleurs, on peut s'assurer qu'un système est canonique en vérifiant que l'algorithme glouton est optimal pour toutes les sommes inférieures à la somme des deux pièces de valeurs maximales.

```
def rendu_glouton(restant, pieces):
    # pieces tableau de valeurs de pièces disponibles dans l'ordre croissant
    indice_pieces = len(pieces) - 1
    rendu = []
    while restant > 0 and  indice_pieces >= 0:
        if pieces[indice_pieces] <= restant:
            restant = restant - pieces[indice_pieces]
            rendu.append(pieces[indice_pieces])
        else:
            indice_pieces = indice_pieces - 1
    # rendu possible
    return rendu

systeme_euro = [1, 2, 5, 10, 20, 50, 100, 200, 500]
assert rendu_glouton(49, systeme_euro) == [20, 20, 5, 2, 2]
```