

Algorithme glouton rendre la monnaie

Christophe Viroulaud

Première - NSI

Algo

Il existe des machines automatiques qui peuvent rendre la monnaie.



FIGURE 1 – Caisse automatique dans une boulangerie

Approche
exhaustive

Approche
gloutonne

Principe
Propriétés
Implémentation

Quel algorithme mettre en place pour rendre la monnaie
en utilisant le moins de pièces possibles ?

Remarques

- ▶ Nous limiterons le problème aux valeurs entières.
- ▶ Nous n'utiliserons que les billets ou pièces de 1, 2, 5, 10€.
- ▶ On supposera que la caisse contient une quantité illimitée de monnaie.

1. Approche exhaustive

2. Approche gloutonne

Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation

Il s'agit de tester toutes les combinaisons possibles avec les pièces dont nous disposons.

Rendre 8€

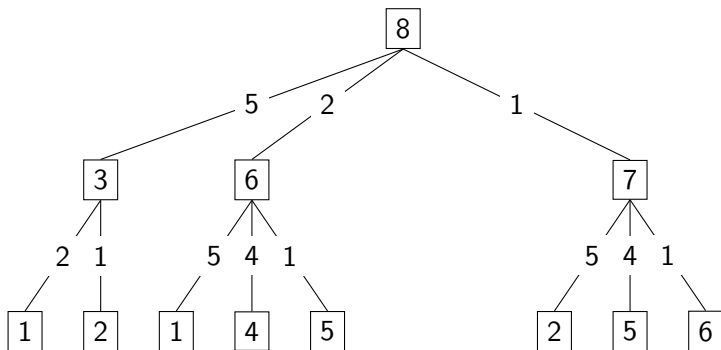
Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation



Approche
exhaustive

Approche
gloutonne

Principe
Propriétés
Implémentation

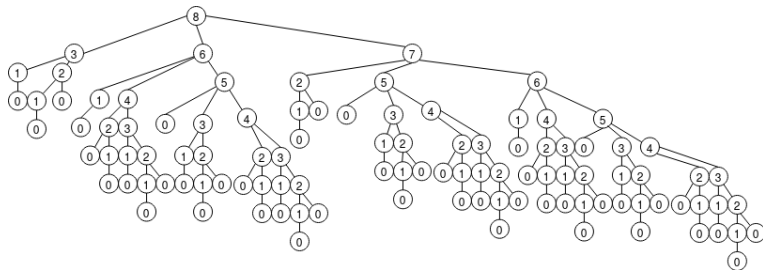


FIGURE 2 – Arbre des possibilités

Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation

1. Approche exhaustive

2. Approche gloutonne

2.1 Principe

2.2 Propriétés

2.3 Implémentation

Approche gloutonne - principe

Pour éviter d'énumérer toutes les solutions possibles, on peut décider un seul choix à chaque étape et ne pas revenir dessus.

À retenir

Un algorithme glouton construit un résultat global par une succession de choix intermédiaire donnant systématiquement le meilleur résultat partiel.

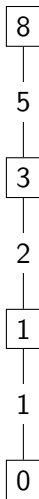


FIGURE 3 – On rend la plus grande pièce possible.

1. Approche exhaustive

2. Approche gloutonne

2.1 Principe

2.2 Propriétés

2.3 Implémentation

Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation

Une approche gloutonne est :

- ▶ plus rapide qu'une solution exhaustive,

Une approche gloutonne est :

- ▶ plus rapide qu'une solution exhaustive,
- ▶ plus simple à implémenter,

Une approche gloutonne est :

- ▶ plus rapide qu'une solution exhaustive,
- ▶ plus simple à implémenter,
- ▶ **ne donne pas forcément la meilleure solution**

L'ancien système britannique est composé de pièces : 1, 3, 4.
Pour rendre 6, l'algorithme glouton propose : 4, 1, 1.
Alors qu'une autre solution optimale existe : 3, 3

Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation

1. Approche exhaustive

2. Approche gloutonne

2.1 Principe

2.2 Propriétés

2.3 Implémentation

```
1  systeme = [10, 5, 2, 1]
```

Code 1 – On représente le système monétaire par un tableau.

Activité 1 : Écrire la fonction `rendu(somme: int, systeme: list) → list` qui renvoie le tableau des pièces à choisir pour rendre `somme`. La fonction implémentera l'algorithme suivant :

Tant que la somme à rendre n'est pas nulle :

- ▶ sélectionner la plus grande pièce possible,
- ▶ la stocker dans un tableau,
- ▶ la soustraire à la somme à rendre.

Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation

```
1 def rendu_monnaie(somme: int, systeme: list) -> list:
2     res = []
3     while somme != 0:
4         # choisir la pièce la plus grande possible
5         i_piece = 0
6         while systeme[i_piece] > somme:
7             i_piece += 1
8         # stocker
9         res.append(systeme[i_piece])
10        # soustraire
11        somme -= systeme[i_piece]
12    return res
13
14 # appel de la fonction
15 rendu_monnaie(14, [10, 5, 2, 1])
```

Approche
exhaustive

Approche
gloutonne

Principe

Propriétés

Implémentation

Code 2 – possibilité 1

```
1 def rendu_monnaie(somme: int, systeme: list) -> list:
2     res = []
3     i_piece = 0
4     while somme > 0:
5         # si pièce est trop grande
6         if systeme[i_piece] > somme:
7             # on avance dans le système
8             i_piece += 1
9         else:
10            res.append(systeme[i_piece])
11            somme -= systeme[i_piece]
12    return res
13
14 # appel de la fonction
15 rendu_monnaie(14, [10, 5, 2, 1])
```

Approche
exhaustive

Approche
gloutonne

Principe
Propriétés
Implémentation

Code 3 – possibilité 2