Gloutonner

G. Aldon - J. Germoni - J.-M. Mény

mars 2012

Algorithme glouton

Le principe de l'algorithme glouton (greedy algorithm) :
faire toujours un choix localement optimal dans l'espoir que ce choix mènera à une solution globalement optimale.

Le problème de la monnaie

On dispose de pièces de monnaie dont les montants sont des nombres entiers de l' unité de monnaie : $t_0 > t_1 > t_2 > \cdots > t_p$.

Pour payer une somme S (entier naturel quelconque), on aimerait utiliser le nombre minimal de pièces possibles.

Choix glouton

On donne la plus "grosse" pièce possible.

Puis à nouveau la plus grosse pièce possible pour le montant restant.

Puis à nouveau la plus grosse pièce possible pour le montant restant.

etc . . .

Choix glouton

On donne la plus "grosse" pièce possible.

Puis à nouveau la plus grosse pièce possible pour le montant restant.

Puis à nouveau la plus grosse pièce possible pour le montant restant.

etc . . .

Programmer

Programmation xcas

La fonction a pour paramètres la somme à payer et la liste T des montants des pièces disponibles. On suppose que les éléments de T sont rangés en ordre décroissant.
La fonction retourne la liste N des nombres de pièces utilisées suivant le principe glouton imposé.

Xcas

```
MONEY(sommeapayer,T) :={
local N,k ;
N :=makelist(0,0,dim(T)-1) ;
pour k de 0 jusque dim(T)-1 faire
N[k] :=iquo(sommeapayer,T[k]) ;
sommeapayer :=irem(sommeapayer,T[k]) ;
fpour ;
retourne(sommeapayer,N) ;} : ;
```

Programmation python

Python

```
def money(somme,ListeMontants) :
    ListeNbPieces=[0 for x in ListeMontants]
    for k in range(len(ListeMontants)) :
        ListeNbPieces[k]=somme//ListeMontants[k]
        somme=somme%ListeMontants[k]
    return somme,ListeNbPieces
```

Choix optimal ?

- L'algorithme mène-t-il toujours à payer avec un nombre minimal de pièces ?
- Plus précisément, peut-on faire un choix pour les montants disponibles :
  - pour lequel l'algorithme n'est pas optimal pour toutes les sommes ?
  - pour lequel l'algorithme est optimal (quelle que soit la somme à payer) ?

Choix optimal ?

On suppose que la liste des montants est : $T := [18, 7, 1]$, c'est à dire que l'on peut disposer de pièces de $t_0 = 18$ €, de pièces de $t_1 = 7$ €et de pièces de $t_3 = 1$€.

L'algorithme conduit-il toujours à payer avec le nombre minimal de pièces ?

Choix optimal ?

On suppose que la liste des montants est : $T := [18, 7, 1]$, c'est à dire que l'on peut disposer de pièces de $t_0 = 18$ €, de pièces de $t_1 = 7$ €et de pièces de $t_3 = 1$€.

L'algorithme conduit-il toujours à payer avec le nombre minimal de pièces ?

Non. Pour une somme à payer de 21€, l'algorithme choisit une pièce de 18€suivie de trois pièces de 1€.
Soit 4 pièces alors que trois pièces de 7€suffisent.

Un choix standard

On suppose que la liste des montants disponibles est $T := [10, 5, 2, 1]$.
Optimalité du glouton ?

Un choix standard

On suppose que la liste des montants disponibles est $T := [10, 5, 2, 1]$.
Optimalité du glouton ?

Oui.

10, 5, 2, 1

Une solution optimale :

10, 5, 2, 1

Une solution optimale :

- utilise au plus une pièce de 5€(sinon on remplace 2 pièces de 5€par une pièce de 10€),

10, 5, 2, 1

Une solution optimale :

- utilise au plus une pièce de 5€(sinon on remplace 2 pièces de 5€par une pièce de 10€),
- au plus 2 pièces de 2€(sinon on remplace 3 pièces de 2€par une de 5€+ une de 1€).

10, 5, 2, 1

Une solution optimale :

- utilise au plus une pièce de 5€(sinon on remplace 2 pièces de 5€par une pièce de 10€),
- au plus 2 pièces de 2€(sinon on remplace 3 pièces de 2€par une de 5€+ une de 1€).
- au plus 1 pièce de 1€(sinon on remplace 2 pièces de 1€par une pièce de 2€).

10, 5, 2, 1

- La somme payée avec les pièces de 1€, 2 €, 5€est donc d'au plus $1 + 2 \times 2 + 5 = 10$€.

10, 5, 2, 1

- La somme payée avec les pièces de 1€, 2 €, 5€est donc d'au plus $1 + 2 \times 2 + 5 = 10$€.
- Elle ne vaut pas 10€(sinon on remplace par une pièce de 10€).

10, 5, 2, 1

- La somme payée avec les pièces de 1€, 2 €, 5€est donc d'au plus $1 + 2 \times 2 + 5 = 10$€.
- Elle ne vaut pas 10€(sinon on remplace par une pièce de 10€).
- Elle vaut donc au plus 9€.
- Le nombre de pièces de 10€utilisées est donc égal à $\lfloor \frac{n}{10} \rfloor$, c'est à dire le nombre calculé par l'algorithme glouton.

10, 5, 2, 1

Pour payer le reste, c'est à dire $n := \mathrm{mod}(n, 10)$, une solution optimale payera avec des pièces de 1€et 2€une somme d'au plus $1 + 2 \times 2 = 5$€. Cette somme est en fait < 5€(sinon ...) et le nombre de pièces de 5€utilisées est donc $\lfloor \frac{n}{5} \rfloor$.

De même pour le nombre pièces de 2€.

Montants en progression géométrique

Et avec $T := [p^k, p^{k-1}, \ldots, p^2, p, 1]$ où $p \in \mathbb{N}$, $p \geqslant 2$?

Montants en progression géométrique

Et avec $T := [p^k, p^{k-1}, \ldots, p^2, p, 1]$ où $p \in \mathbb{N}$, $p \geqslant 2$?

Optimal à nouveau.

$p^k, p^{k-1}, \ldots, p^2, p, 1$

Une solution optimale utilise :

- au plus $p-1$ pièces valant p^{k-1} (sinon on en remplace p par une pièce de p^k).

$p^k, p^{k-1}, \ldots, p^2, p, 1$

Une solution optimale utilise :

- au plus $p-1$ pièces valant p^{k-1} (sinon on en remplace p par une pièce de p^k).
- au plus $p-1$ pièces valant p^{k-2} (sinon on en remplace p par une pièce de p^{k-1}).
- ...

$p^k, p^{k-1}, \ldots, p^2, p, 1$

Une solution optimale utilise :

- au plus $p-1$ pièces valant p^{k-1} (sinon on en remplace p par une pièce de p^k).
- au plus $p-1$ pièces valant p^{k-2} (sinon on en remplace p par une pièce de p^{k-1}).
- ...

La somme payée avec les pièces de 1, p, ..., p^{k-1} est donc d'au plus :

$$(p-1)(1+p+p^2+\cdots+p^{k-1}) = (p-1) \times \frac{p^k-1}{p-1} = p^k - 1$$

$p^k, p^{k-1}, \ldots, p^2, p, 1$

le nombre de pièces de p^k €utilisées dans une solution optimale pour payer n €est donc $\lfloor \frac{n}{p^k} \rfloor$, c'est à dire le nombre déterminé par l'algorithme ...